Universidad
Carlos III de Madrid

Departamento de Informática

PROYECTO FIN DE CARRERA

INGENIERIA TÉCNICA DE TELECOMUNICACIÓN:
SONIDO E IMAGEN

Reconocimiento Automático de Voz en APEINTA desde Terminales Móviles

Autor: Álvaro Martín Nogales

Tutor: Ana María Iglesias Maqueda

Leganés, Octubre de 2015

TÍTULO: *Reconocimiento automático de voz en APEINTA desde terminales móviles.*

AUTOR: *Álvaro Martín Nogales*

TUTOR: Ana María Iglesias Maqueda

DIRECTOR:

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos.

En primer lugar, quiero agradecer al CESyA la oportunidad hace 5 años de dejarme comenzar esta aventura con ellos. Desde el primer día me facilitaron mucho las cosas. Gracias.

En segundo lugar a mi tutora Ana María, por todos estos años de paciencia en mis 'idas y venidas' para retomar el PFC. Siempre te agradeceré la disponibilidad y la confianza depositada en mí.

En tercer lugar, a Juan Francisco, compañero y guía durante mi colaboración con el CESyA. Muchas gracias por el apoyo técnico y tu disponibilidad. También al resto de compañeros del CESyA: Diego, Javi, Pachi, Alberto, Juan Manuel...

A todos mis amigos en general, especial mención para Gago, Dela, Carlos, Dani, Bieito, Juan, Kike, Locu, Raulito, Bellon, Carlitos, Almu, María, Carol, Jesús, Roberto, Pedro, Juanmi...sois muchos y no puedo mencionaros a todos.

Una línea aparte para Diego, él sabe por qué. Muchas gracias.

Gracias Rafa por tu apoyo con esto. Durante estos años siempre has estado ahí para lo que ha hecho falta.

A mi primo y compañero durante la carrera, Jorge. Me has empujado constantemente a no abandonar el PFC y en general, a todas mis metas. Eres como un segundo hermano para mí.

A mi ex-pareja, Natalia. Por todo este tiempo que estuvimos juntos y conté con tu apoyo incondicional para el PFC y para todo en general. Eres de alguna manera parte de todo esto...No lo olvidaré.

A Erik. Una persona con una paciencia a raudales. Agradecerte desde lo más profundo todos estos años, especialmente estos 3 últimos meses. Gracias por absolutamente todo.

A mi hermano Daniel, sobran los motivos...

Pero sobre todo gracias a mis padres. Si hoy estoy escribiendo estas líneas es por ellos. Este proyecto supone el final de una etapa de mi vida y ha supuesto un gran esfuerzo. Es un regalo para mí mismo pero también para vosotros...Os quiero.

Resumen.

El subtítulado en directo para entornos educativos puede transformar la experiencia de estudiantes con discapacidad auditiva en edad escolar o universitaria, es decir, en pleno desarrollo intelectual, al permitir su plena participación en las clases regulares. Por esta razón, mejorar la precisión del reconocimiento de voz debe ser un objetivo prioritario de la comunidad científica a pesar de que en cualquier transcripción hay inevitablemente palabras que siempre se transcriben de forma incorrecta.

La operadora de Telecomunicaciones France Télécom España, S.A, solicitó a la Universidad Carlos III de Madrid realizar un estudio de viabilidad acerca de la posibilidad de enviar voz desde un terminal móvil en tiempo real hasta un servidor donde se procesaría el audio para obtener transcripciones a texto.

El Centro Español de Subtitulado y Audiodescripción, en colaboración con la Universidad Carlos III, ha decidido incluir este estudio dentro del contexto de su proyecto APEINTA, en vista a las posibles mejoras que este nuevo estudio podría generar en la evolución del proyecto.

APEINTA es un sistema de subtitulado automático, en directo y diferido, que utiliza la tecnología del reconocimiento automático del habla para transcribir la voz a texto, de modo que los alumnos con discapacidad auditiva puedan seguir sin dificultad las explicaciones del profesor al leer únicamente texto a través de algún medio visual como smartphones o tabletas.

Hasta ahora, la voz del profesor se grababa con un micrófono conectado directamente al servidor de subtitulado de APEINTA. Existe, por tanto, una necesidad de establecer un mecanismo de comunicación alternativo al que actualmente existe entre el profesor y el servidor de subtitulado, con el objetivo de proporcionar soluciones de movilidad para el profesor independientemente de su localización.

El presente proyecto fin de carrera se centrará en el estudio de soluciones para precisamente satisfacer esta necesidad y así ofrecer más alternativas de despliegue para el proyecto APEINTA, como la posibilidad de transmitir la voz desde un dispositivo Android por streaming hasta el servidor de subtitulado, el cual utiliza Dragon Naturally Speaking como motor de reconocimiento de voz.

Abstract.

Live captioning for educational environments can change the experience of students with temporary or permanent hearing impairment at school or college, allowing them to participate in regular classes. For this reason, to improve the voice recognition accuracy in automatic speech recognition systems should be a priority for the scientific community.

Some years ago, the telecommunications operator France Telecom Spain requested a viability study to Universidad Carlos III of Madrid about the possibility of sending voice from a mobile terminal to a server where audio would be processed in real time to obtain text transcriptions. This final project is focused on carrying out the viability study.

The Spanish Center for Captioning and Audio description, in collaboration with Universidad Carlos III of Madrid, decided to include this study in the context of APEINTA project, in view of the possible improvements which could lead to APEINTA deployment project for the future.

APEINTA is an automatic captioning service, live and recorded, which uses the automatic speech recognition technology in order to transcribe voice to text, so students with hearing disabilities can follow the teacher's explanations without difficulty. To do this, they can read text through a visual medium such as smartphones, tablets, personal computers etc.

Before the study presented in this document, teachers needed to be physically connected to the APEINTA server via a microphone in order to capture the audio with enough quality. There is therefore a need to design an alternative communication mechanism between the teacher and the real-time transcription server, with the main objective of providing mobility access to the teacher regardless of their location.

This project will focus on the study of solutions to find more alternatives for APEINTA deployment project, as the possibility of transmitting voice from an Android device to the transcription server, which uses Dragon Naturally Speaking as voice recognition engine.

Índice General.

Índice de Figuras

Índice de Tablas

Capítulo 1. Introducción y Objetivos

1.1 Planteamiento del problema

El estudio e investigación de nuevas tecnologías para la accesibilidad audiovisual es una de las claras propuestas a abordar en nuestra sociedad actual. En este contexto, el proyecto sugerido por el Centro Español de Subtitulado y Audiodescripción (CESyA) [1] conocido como APEINTA (Apuesta por la Enseñanza Inclusiva dentro y fuera del Aula) [2], el cual se explicará con detalle en el capítulo 3.2, pretende encontrar soluciones usando las nuevas tecnologías de la información para frenar las barreras existentes en el acceso a la educación.

En este sentido, el proyecto APEINTA apuesta por la educación inclusiva dentro y fuera del aula para todos los estudiantes, independientemente de si éstos presentan o no algún tipo de discapacidad. Se basa en dos propuestas inclusivas:

- **Propuesta 1**: Se enfoca su desarrollo dentro del aula. Para ello se utilizan dos mecanismos para tratar de eliminar los problemas de comunicación presentes en las aulas: mecanismos de reconocimiento automático del habla (ASR: *Automatic Speech Recognition*) para una trascripción en tiempo real, orientada a personas que tengan discapacidad auditiva temporal o permanente, y mecanismos de síntesis de voz (TTS: *Text To Speech*) para facilitar la comunicación oral entre el profesor y el alumno.
- **Propuesta 2**: Centrada en la educación inclusiva fuera del aula. Los estudiantes disponen de una plataforma accesible de enseñanza Web con recursos digitales a los que pueden acceder en todo momento, independientemente del lugar.

Este proyecto fin de carrera surgió como una iniciativa de investigación de otro proyecto firmado por la Universidad Carlos III [3] y financiado por France Telecom España el 28 de mayo de 2010 titulado "Adaptación de la Herramienta informática APEINTA a dispositivos cliente eReader y conexiones IP", el cual fijaba como objetivo mejorar las prestaciones de usabilidad de APEINTA a través del uso de nuevas tecnologías móviles.

Actualmente, el servicio de APEINTA es capaz de proporcionar acceso mediante un dispositivo móvil a los estudiantes, es decir, los estudiantes pueden conectarse al servidor de subtitulado sin importar su ubicación, pero los profesores necesitan establecer una conexión física con el servidor a través de un micrófono con el fin de capturar el audio con la calidad suficiente para un correcto reconocimiento de voz.

Por tanto, se tratará a lo largo de los sucesivos capítulos de mejorar el mecanismo de comunicación entre el maestro y el servidor de subtitulado en tiempo real, analizando si es posible facilitar el acceso del profesor al sistema mediante el uso de nuevas tecnologías de comunicación como voz sobre IP (VoIP), entendiendo este concepto para este estudio como el envío genérico de la voz sobre redes IP, es decir, sin ningún protocolo de señalización específico asociado. En concreto, el estudio se centrará en la transmisión de voz desde smartphones con Android a través de redes móviles de datos o redes inalámbricas hasta el servidor de transcripción de APEINTA y una vez recibido, abordar el tratamiento del flujo de voz para realizar la conversión a texto.

1.2 Objetivos

El actual sistema de subtitulado en directo desarrollado para entornos educativos por el CESyA, base de la que parte este proyecto fin de carrera, ha demostrado ser una buena arquitectura sobre la que integrar el acceso desde diferentes dispositivos para alumnos con discapacidades auditivas, de modo que se mejoren los niveles de accesibilidad para este grupo. De esta forma, la arquitectura del servicio de transcripción en tiempo real de APEINTA permite a los estudiantes utilizar diferentes dispositivos para la lectura de la transcripción a texto de la voz del profesor [4]. Por otra parte, este servicio se proporciona a los estudiantes independientemente de su ubicación (puesto que no necesitan estar dentro de las aulas para el acceso a la transcripción).

Sin embargo, el número de dispositivos que puede manejar el profesor actualmente dentro de dicho sistema no resulta tan amplio, limitándose al uso de un micrófono convencional conectado al propio servidor de transcripción. Un micrófono de buena calidad y su conexión directa con el servicio garantiza una buena calidad de audio y por lo tanto, una buena calidad en la transcripción mediante ASR.

Con este proyecto fin de carrera se estudiarán otros mecanismos de comunicación alternativos para el profesor con el fin de aumentar el número de posibilidades técnicas de acceso al servicio de transcripción en tiempo real. Más en concreto, el principal análisis se centrará en verificar la viabilidad del uso de otra serie de dispositivos para la comunicación profesor-servidor, como los teléfonos móviles de última generación, evitando la dependencia directa de los micrófonos convencionales.

Por este motivo, además del auge de los nuevos terminales móviles y sus sistemas operativos, con navegadores web totalmente funcionales y conexión de banda ancha a internet, se decidió que, para facilitar la recepción de audio y su posterior procesamiento para la generación de subtítulos, se podría implementar una nueva arquitectura con un servidor de streaming de audio para teléfonos móviles, el cual tuviera conexión a través de internet al resto del Sistema de APEINTA mediante la tecnología VoIP.

Por tanto, el principal objetivo de este estudio es la valoración de la viabilidad del envío de voz desde un terminal móvil hasta un servidor de subtitulado mediante la tecnología de streaming, ideal para transmisiones de contenido multimedia en tiempo real, a través de una red de datos con el menor retardo posible [5].

Si se tiene en cuenta tanto la actual arquitectura de APEINTA así como el principal objetivo definido anteriormente, parece obvio pensar que solo será necesario sustituir el micrófono por un terminal móvil como alternativa a la captación de la voz. El resto del sistema no se vería alterado.

De esta forma, la voz del profesor será grabada por el terminal móvil que automáticamente enviará el streaming de audio hacia el servidor de subtitulado de APEINTA, el cual generará subtítulos que serán, por último, transmitidos a los dispositivos finales de lectura de subtítulos que manejarán los alumnos. Estos dispositivos ya han sido analizados en proyectos anteriores dentro del contexto de APEINTA, por tanto, no es objetivo de este documento abordar este tema.

Analizando el principal objetivo, es decir, proporcionar acceso móvil al profesor para el servicio de transcripción en tiempo real de APEINTA, surgen de forma inmediata otros más específicos y secundarios que se citarán a continuación:

- Estudio de nuevos dispositivos para la grabación de voz compatibles con el servicio de transcripción en tiempo real de APEINTA, evitando la dependencia directa de un micrófono directamente conectado al servidor de transcripción.
- Evaluar el coste de migrar a otras plataformas de dispositivos móviles. Para el presente proyecto fin de carrera, la prueba de viabilidad se ha llevado a cabo en una única plataforma móvil específica. Sin embargo, este estudio considera también otras plataformas para los diseños y desarrollos futuros.
- Reducir el coste de despliegue (instalación y configuración) de APEINTA. Hoy en día, un teléfono móvil es un dispositivo muy común que cualquier persona es capaz de manejar de forma sencilla por lo que ya no sería necesario el uso de otros dispositivos específicos y más complejos.
- Reducir el coste de acceso al servicio de APEINTA por parte del profesor. Éste sólo necesita un terminal móvil para acceder al servicio utilizando una simple interfaz, independientemente de su ubicación física.

Todos los puntos anteriores describen los objetivos del proyecto y definen sus justificaciones, siempre teniendo en cuenta que el futuro final y deseable de APEINTA se centra en facilitar el acceso a dicho sistema especialmente al alumnado con problemas de discapacidad auditiva. A pesar de ello, no hay que olvidar la búsqueda de alternativas tecnológicas reales y posibles para otras partes del sistema como nuevas formas de comunicación entre el profesor y el servidor de subtitulado.

1.3 Estructura del documento

El presente documento que representa la memoria de este Proyecto Fin de Carrera, contiene una portada, un índice de contenidos, un índice de figuras y tablas, nueve apartados, una bibliografía y anexos.

A continuación se detallan los nueve apartados principales:

- **Introducción y objetivos**: contiene la motivación o planteamiento del problema, los objetivos del proyecto, y una descripción de la estructura de la memoria.
- **Estado del arte**: estudio sobre los principales sistemas operativos móviles, el subtitulado en tiempo real, algunos sistemas para realizar subtitulado en entornos educativos o con tecnología móvil, así como una descripción del marco regulador relacionado con estas tecnologías.
- **Marco de trabajo**: se realiza un estudio sobre el sistema APEINTA y el cliente SipDroid para Android, que han sido la base de este proyecto.
- **Análisis del proyecto**: contiene la descripción del sistema y el conjunto de requisitos de usuario y software que tiene que cumplir.
- **Diseño del proyecto**: explica la relación entre los requisitos de análisis y los requisitos funcionales de aplicación, detallando el diseño realizado para cada uno.
- **Implementación**: se realiza un estudio comparativo de la codificación y streaming de la voz, para justificar la arquitectura final a implementar y los detalles de desarrollo del sistema cliente-servidor.
- **Pruebas**: se detalla la batería de pruebas realizadas para demostrar la viabilidad del proyecto dentro del contexto de APEINTA.
- **Gestión del proyecto**: se especifica la metodología de trabajo, de desarrollo y las fases de realización del proyecto. Además, se realiza un estudio del presupuesto requerido.
- **Conclusiones y trabajos futuros**: se explican todas las conclusiones derivadas del proyecto y se proponen nuevas ideas para continuar con la línea de desarrollo del mismo.

Capítulo 2. Estado del Arte

2.1 Sistemas operativos móviles

2.1.1 Introducción

Es indiscutible la evolución que han seguido los smartphones a lo largo de las dos últimas décadas, posibilitando a cualquier persona disponer de los últimos avances tecnológicos y transformando las comunicaciones interpersonales.

De esta forma, por ejemplo Apple consiguió crear con su iPhone OS un nuevo sistema operativo que supuso una clara innovación en el ámbito del software para dispositivos móviles con respecto a plataformas como Symbian o Windows Mobile, las dos más populares hasta ese momento. Ante esta nueva situación en el año 2007, algunas empresas como Microsoft o Nokia decidieron actualizar sus sistemas operativos, y otras comenzaron a desarrollarlos partiendo de cero, como Google con Android o Nokia con Maemo. En cualquier caso, el nuevo terminal iPhone alcanzó un número de ventas realmente relevante, debido principalmente a su pantalla táctil única y a un evolucionado navegador web.

En la actualidad, Google ejerce su total hegemonía en el mercado de los smartphones a nivel mundial con cerca de un 83% de cuota de mercado según IDC [6] (*International Data Corporation*), es decir, ocho de cada diez dispositivos aproximadamente tiene Android como sistema operativo (ver Tabla 1). Android es elegido prácticamente por la totalidad de los fabricantes de smartphones como sistema operativo, salvo Windows Phone, que ha adquirido cierta importancia entre otros fabricantes como HTC independientemente de que su impulsor fundamental siga siendo Microsoft/Nokia, y los sistemas de Apple y Blackberry que no están disponibles para otras marcas.

Periodo	Android	iOS	Windows Phone	BlackBerry OS	Otros
2015Q2	82.8%	13.9%	2.6%	0.3%	0.4%
2014Q2	84.8%	11.6%	2.5%	0.5%	0.7%
2013Q2	79.8%	12.9%	3.4%	2.8%	1.2%
2012Q2	69.3%	16.6%	3.1%	4.9%	6.1%

Tabla 1. Cuota de mercado de los sistemas operativos móviles.

Como puede deducirse de la Tabla 1, existen varias plataformas que continúan creciendo a pesar de que tan sólo supongan un 0.4% del total. La mayoría de ellas están basadas en código libre y en este grupo se pueden incluir Tizen, Sailfish OS, Ubuntu Phone o Firefox OS.

Con el objetivo de justificar mejor la elección del sistema operativo para la aplicación móvil que se ha desarrollado, se detalla a continuación un estado del arte acerca de las plataformas móviles más importantes y sus características técnicas, dando más importancia a aquellas que posean una importante cuota de mercado en la actualidad.

2.1.2 Android

Figura 1. Logotipo Android

2.1.2.1 Historia

Android [7] es un sistema operativo para dispositivos móviles inicialmente desarrollado por Android Inc., una pequeña empresa comprada por Google [8] en el 2005. Se basa en una versión modificada del Kernel de Linux. Es un producto mantenido por la *Open Handset Alliance* (OHA) [9], una alianza comercial de 78 compañías de hardware, software y telecomunicaciones dedicadas al desarrollo de estándares abiertos para dispositivos móviles.

Como ya se ha mencionado, Google adquirió Android Inc., una pequeña startup con base en California en Julio de 2005. A partir de ese momento, comenzaron los rumores de que Google estaba planeando entrar en el mercado de la telefonía móvil con esta adquisición.

En septiembre del 2007 se tuvo constancia de que Google había patentado una serie de aplicaciones en el área de telefonía móvil gracias a una publicación de *InformationWeek*. El 5 de noviembre del mismo año, se anunció la fundación de la OHA, y también se dio a conocer su primer producto: Android. Un año más tarde, el 9 de diciembre de 2008, la OHA abriría sus puertas a 14 nuevos miembros que se añadirían al proyecto Android.

El primer teléfono en el mercado que funcionó con Android fue el T-Mobile G1 (también conocido como *Dream*), lanzado el 22 de octubre de 2008 con la Android 1.0 preinstalado.

Desde el 21 de octubre de 2008, Android está disponible como código abierto. Gracias a ello, cualquiera puede añadir extensiones, nuevas aplicaciones o reemplazar las existentes por otras dentro del dispositivo móvil.

En la actualidad existen más de 1.500.000 aplicaciones para Android y más de un millón de teléfonos móviles se activan diariamente. Se trata de la plataforma móvil con más penetración en el mercado acaparando el 80% del total.

La última versión estable es la 5.1.1 cuyo nombre es *Lollipo* y fue liberada el 21 de Abril de 2015. Sin embargo, ya existe una versión preliminar de Android 6.0, denominada *Marshmallow*, publicada el 17 de agosto de 2015.

2.1.2.1.1 Historial de Actualizaciones

Con la intención de corregir *bugs* y añadir nuevas funcionalidades, Android ha sufrido numerosas actualizaciones sobre su sistema operativo base, incluyendo el kernel de Linux, desde su lanzamiento inicial en Septiembre de 2008.

El siguiente gráfico se basa en el número de dispositivos Android que han tenido acceso al *Google Play Store* [10], el cual únicamente soporta Android 2.2 o superior, durante los 7 días anteriores al 7 de Septiembre de 2015:

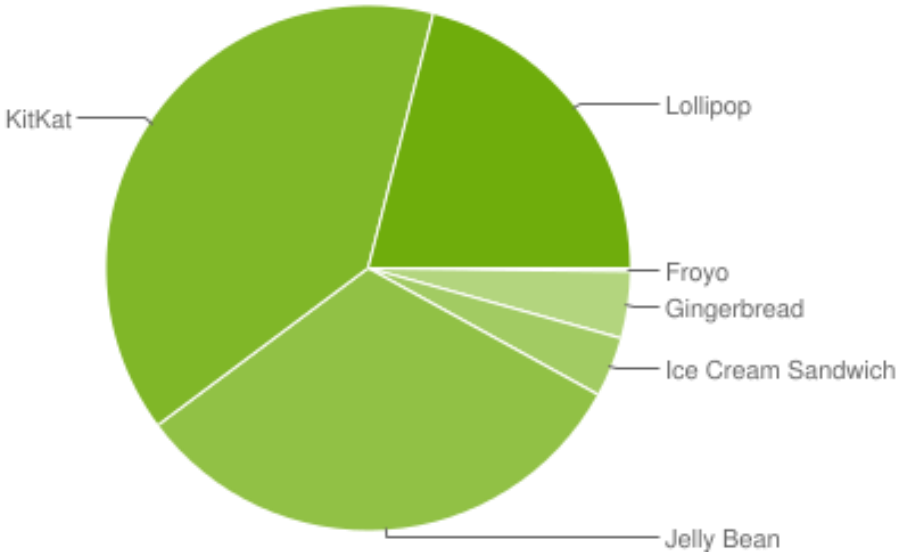

Figura 2. Distribución de dispositivos Android en función de su versión.

Se puede apreciar claramente la hegemonía actual de las versiones 4.1-4.3 (*Jelly Bean*) con un 31,8%, 4.4 (*KitKat*) con un 39,2% y 5.0-5.1 (*Lollipop*) con un 21% sobre el resto de las disponibles en el mercado.

2.1.2.2 Características

2.1.2.2.1 Interfaz gráfica

La interfaz gráfica de la última versión de Android (*Lollipop*) ofrece un rediseño muy diferente respecto a las versiones anteriores, construido alrededor del concepto de *Material Design* como nuevo lenguaje de diseño, siendo especialmente apta para la lectura en pantallas de alta resolución como se puede apreciar en la siguiente figura:

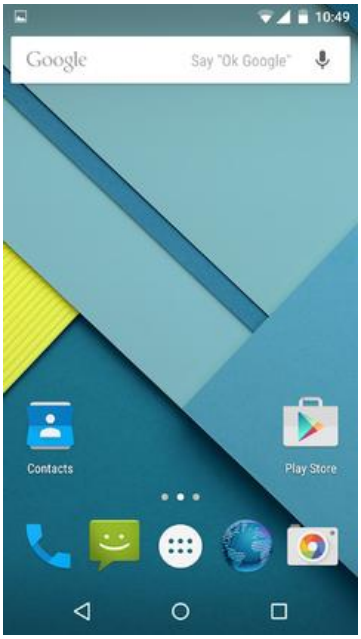

Figura 3. Interfaz gráfica de Android 5.0.

Los botones físicos que típicamente aparecían en los terminales con Android antiguos se sustituyen por nuevos botones de acción visuales al igual que ocurre con los círculos y formas flotantes en relación a los botones deslizantes. En la nueva pantalla de desbloqueo se puede acceder inmediatamente a funciones como la cámara o las notificaciones, e incluso obviar avisos con un deslizamiento del dedo. Las notificaciones también se muestran dentro de las aplicaciones como banners situados en la parte superior de la pantalla.

También existe una pantalla de Favoritos, accesible desde cualquier otra pantalla, en la cual se pueden añadir los *widgets*. Los *widgets* resultan ser muy dinámicos ampliando su funcionalidad tradicional hasta resultar casi aplicaciones independientes: leer mails, consultar el calendario, escuchar música o visualizar actualizaciones de redes sociales.

2.1.2.2.2 Arquitectura

El diagrama de la figura 4 muestra los componentes principales de Android desde el punto de vista de su arquitectura. Cada sección se describe con más detalle a continuación:

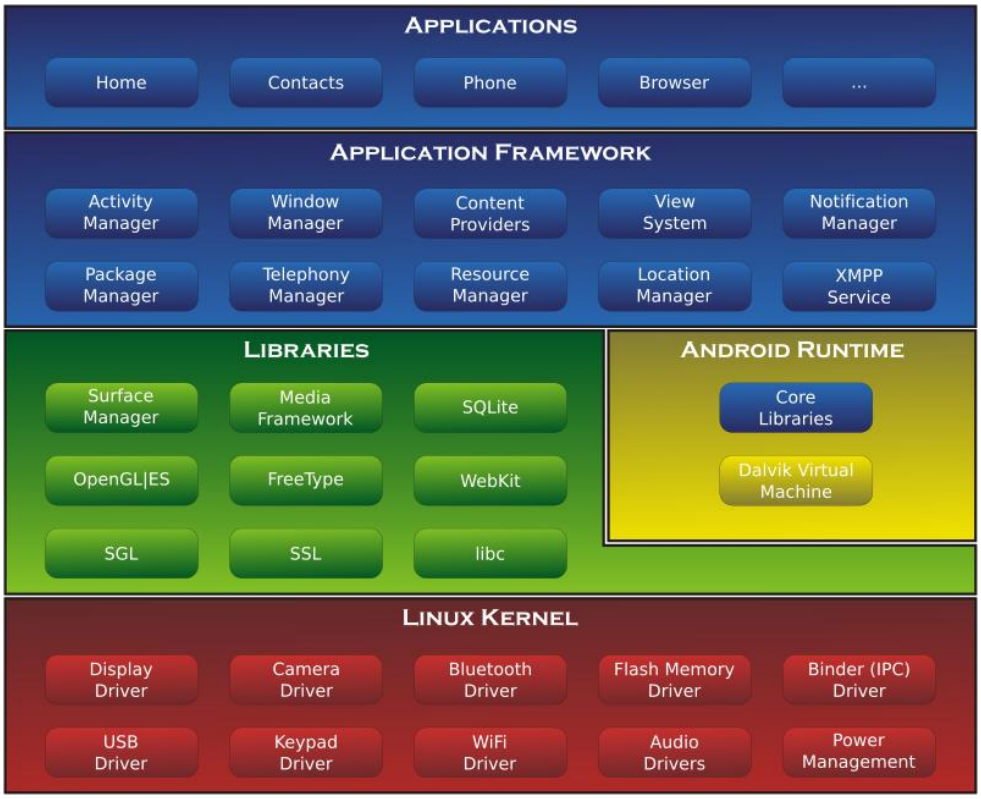

Figura 4. Arquitectura de la plataforma Android.

Aplicaciones: Android ofrece una serie de aplicaciones por defecto para cualquier dispositivo tales como un cliente de email, un programa de SMS (servicio de mensajería corta), calendario, mapas, navegador web, contactos, y otros. Habitualmente están escritas en Java, aunque pueden usarse de forma nativa otros lenguajes de programación.

Framework de aplicaciones: Al proporcionar una plataforma de desarrollo abierto, este framework es ofrecido en forma de gran cantidad de APIs (Interfaz de programación de aplicaciones) disponibles en el SDK (Herramientas de desarrollo de software) para que el desarrollador pueda crear aplicaciones muy funcionales totalmente nuevas.

Librerías: La mayor parte de los componentes del sistema utilizan un set de librerías escritas en C/C++. El framework de aplicaciones se comunica con estas librerías mediante JNI (Interfaz nativa de Java) y en la mayor parte de las ocasiones resulta transparente para el desarrollador.

Runtime de Android: La máquina virtual Dalvik ha sido diseñada específicamente para esta plataforma y se caracteriza por consumir poca memoria y recursos de manera que cada aplicación Android se ejecuta en una instancia propia de esta máquina. A su vez, cada aplicación corre en su propio proceso, delegando al kernel la gestión de memoria a bajo nivel e hilos. Sin embargo, en Android 4.4 se introdujo ART (*Android Runtime*) de forma experimental como una alternativa a Dalvik adoptándose de forma predeterminada en la versión 5.0.

Kernel de Linux: Gestiona los servicios más básicos del sistema como seguridad, gestión de memoria, gestión de procesos, stack de red, y modelo de controladores. También puede verse como una capa de abstracción entre el hardware y el software.

2.1.2.3 Desarrollo

Android tiene una de las mayores comunidades de desarrolladores y dispone de una tienda de aplicaciones online administrada por Google: *Google Play* (anteriormente *Android Market*). Ésta retribuye el 70% del precio de la aplicación a sus desarrolladores.

Una de las grandes ventajas de Android es que el código de las aplicaciones se escribe en Java, el lenguaje de programación multiplataforma más extendido, a través del SDK proporcionado por el mismo Google de forma gratuita. Este SDK, soportado en Windows, Linux y Mac, incluye un conjunto de herramientas de desarrollo típicas como un depurador, un emulador (basado en QEMU [11]), librerías y documentación con tutoriales.

Inicialmente, el IDE (entorno de desarrollo integrado) oficialmente soportado era Eclipse [12] junto con el plugin ADT (herramientas de desarrollo de Android) [13]. Sin embargo, en diciembre de 2014 Google anunció Android Studio [14] como su IDE oficial.

Otras alternativas son *Google App Inventor*, una plataforma de Google Labs con un entorno visual para programadores novatos, y *Apache Cordova* [15], que permite crear aplicaciones nativas para Android utilizando lenguajes de programación web. También se pueden usar otros lenguajes, como C o C ++, mediante el uso del NDK (Kit de desarrollo nativo) [16] de Android.

Desde Septiembre del 2008 Google ha publicado nuevas actualizaciones del SDK constantemente. La última es la versión 24.

2.1.2.3.1 Licencia

Android se desarrolla de forma abierta desde el 21 de octubre de 2008 bajo la licencia de Apache (*Apache Software License* 2.0) [17] por lo que el código fuente es accesible completamente. Además, con esta licencia, los desarrolladores pueden añadir extensiones propietarias, sin necesidad de que las mismas también se liberen bajo la misma licencia.

Sin embargo, es importante recordar que el código para soportar el hardware (controladores) de cada fabricante habitualmente no es público y por ello, no todos los terminales con Android funcionan con la última versión de la plataforma.

2.1.3 Blackberry OS

Figura 5. Logotipo BlackBerry.

2.1.3.1 Historia

BlackBerry OS es un sistema operativo móvil desarrollado por la empresa canadiense BlackBerry [18], inicialmente RIM (Research In Motion), para sus dispositivos propietarios.

RIM nace en 1995 a través de financiaciones de institucionales canadienses y otros inversores. Sus primeras contribuciones al mercado móvil se remontan a 1999 a pesar de que no fue tres años más tarde, en 2002, hasta que no apareció el primer smartphone creado por RIM. Desde entonces, RIM ha sido una compañía en continua expansión potenciando sus operaciones globales y absorbiendo a otras compañías para mejorar las prestaciones de sus teléfonos.

Sin embargo, no fue hasta Enero de 2013 cuando RIM, aprovechando el lanzamiento de BlackBerry 10, cambiaría de nombre para denominarse como su producto más conocido.

La última versión de este sistema operativo es la 10.3, por lo que todo parece indicar que la compañía ha decidido centrarse en el desarrollo y mejora a largo plazo de esta versión.

2.1.3.2 Características

En la última versión, sin dejar a un lado la parte profesional, se presta especial interés al rendimiento, a la parte multimedia y a la interfaz gráfica con un enfoque al usuario doméstico.

Así la versión 10 es un sistema operativo multitarea de código cerrado desarrollado únicamente por BlackBerry por lo que su núcleo es propietario. Está basado en *QNX Neutrino* minimizando la cantidad de procesos en el espacio del kernel y ejecutando el resto en el espacio de usuario permitiendo gestionar los procesos que no responden de forma aislada. Adicionalmente, el sistema operativo realiza una prueba de integridad antes de arrancar.

2.1.3.3 Desarrollo

Las aplicaciones para BlackBerry están disponibles a través de la *BlackBerry App World* [19]. Además en la última versión de la plataforma se permite ejecutar aplicaciones de Android de forma fluida de forma que ya se puede optar a las aplicaciones disponibles en Google Play.

Todas las herramientas de desarrollo son abiertas y gratuitas, multiplataforma y con numerosos plugins para Eclipse, Visual Studio o NetBeans. De esta forma, se dispone de un conjunto de APIs que abarcan desde C y C++, Adobe AIR, AJAX (JavaScript asíncrono y XML) o HTML5 (Lenguaje de Marcado de Hipertexto).

Por ejemplo, *BlackBerry Theme Studio* es una suite gratuita de herramientas de diseño de gráficos y temas para esta plataforma.

2.1.4 iOS.

Figura 6. Logotipo iOS.

2.1.4.1 Historia

iOS, anteriormente conocido como iPhone OS, es un sistema operativo móvil de Apple [20] cuyo desarrollo original estuvo enfocado hacia el único terminal móvil de la compañía hasta el momento, el iPhone.

iPhone OS se dio a conocer, sin un nombre concreto, el 9 de enero de 2007 en la *Macworld Conference & Expo*, aunque su lanzamiento no tuvo lugar hasta unos meses después, coincidiendo con el proceso de comercialización de la primera generación de teléfonos iPhone.

El cambio oficial de nombre de iPhone OS a iOS fue anunciado el 7 de junio de 2010 durante la presentación del iPhone 4. El 12 de septiembre de 2012 se presento el nuevo iPhone 5 junto con las nuevas características de iOS 6 y el 10 de Junio de 2013 la siguiente versión. Actualmente la última actualización disponible es iOS 8.4.

2.1.4.2 Características

iOS se caracteriza por la manipulación de la interfaz de usuario a través de deslizadores, interruptores y botones mediante gestos multitáctiles provocando una respuesta inmediata:

- Barra de estado: situada en la parte superior para mostrar datos típicos.
- *SpringBoard*: es la pantalla principal y ocupa casi toda la superficie.
- *Dock*: situado en la parte inferior para colocar las aplicaciones más frecuentes.

Una de las desventajas de la plataforma es que no tiene soporte para Adobe Flash o Java, aunque si lo hace con HTML5 como alternativa para sitios web con estas tecnologías.

A nivel de arquitectura, el iOS presenta cuatro capas de abstracción: el núcleo (el mismo que para MAC OS X), los servicios primarios o básicos, los servicios multimedia y de generación de gráficos y *Cocoa Touch*, un framework para aplicaciones de usuario.

2.1.4.3 Desarrollo

Desde Marzo de 2008 hay disponible un SDK para terceros y desarrolladores (solo para plataformas MAC), el cual incluye un simulador (*iPhone simulator*) para hacer pruebas de aplicaciones para iPhone, iPod Touch etc, el entorno de desarrollo oficial XCode [21] y el *Interface Builder* para diseñar las interfaces gráficas. El lenguaje de programación habitual es *Objective-C*, basado en las versiones C estándar.

Dados los grandes beneficios que está aportando la tienda de aplicaciones de Apple, conocida como *App Store*, una gran cantidad de desarrolladores han apostado por esta plataforma, lo que ha provocado un crecimiento desorbitado del número de aplicaciones para iPhone.

2.1.5 Windows Phone.

Figura 7. Logotipo Windows Phone.

2.1.5.1 Historia

Windows Phone [22], anteriormente conocido como Windows Mobile, es un sistema operativo móvil desarrollado por Microsoft [23], y está diseñado para ser usado en dispositivos móviles.

La primera versión de Windows Phone, Windows Phone 7 fue lanzada finalmente el 11 de Octubre de 2010. Esta primera versión presentaba muchas carencias por lo que Microsoft, tras su acuerdo con Nokia del 11 de febrero de 2011, continuó liberando actualizaciones como Windows Phone 7.5 (*Mango*) o Windows Phone 7.5.1 (*Tango*). Se trataba de actualizaciones enfocadas a minimizar los requisitos del sistema para adaptarlo a terminales de menor coste.

La segunda versión de Windows Phone, Windows Phone 8, se anunció el 29 de Octubre del 2012 y su última actualización definitiva es Windows Phone 8.1, lanzada el 14 de Abril de 2014.

Finalmente, Microsoft anunció en Enero de 2015 que no continuaría con Windows Phone para centrarse en un único sistema para todo tipo de plataformas denominado Windows 10.

2.1.5.2 Características

Windows 10 está diseñado para smartphones de menos de 8 pulgadas que manejen arquitecturas ARM (Arquitectura RISC de 32 bits) o IA-32 (Arquitectura Intel de 32-bit).

Microsoft ha incluido numerosas modificaciones visibles por el usuario como nuevos menús, transparencias, nueva organización de los elementos y un nuevo sistema de notificaciones. Además, conserva la fluidez y estabilidad de la interfaz de Windows Phone 8. También se ha introducido *Continuum*, un nuevo sistema para ejecutar aplicaciones de escritorio en el smartphone o incluso portar la pantalla del smartphone a un monitor.

Por último, hay soporte completo para Office y su ecosistema de almacenamiento en la nube, y *Edge* sustituye definitivamente a Internet Explorer como explorador web.

2.1.5.3 Desarrollo

El IDE para el desarrollo de aplicaciones es Visual Studio 2015 junto con un SDK [24] independiente de Windows que incluye un editor de código, un depurador eficaz, emuladores de Windows Mobile y un amplio soporte de lenguajes.

Los lenguajes de programación utilizados para el desarrollo de aplicaciones son principalmente Visual C++ y C# o Visual Basic así como el lenguaje de marcado XAML (Lenguaje Extensible de Formato para Aplicaciones) para el desarrollo de las interfaces de usuario.

En lo relativo a la distribución de aplicaciones, Microsoft ya dispone de una tienda online [25].

2.2 Subtitulado en tiempo real

En la actualidad cada vez es más frecuente adaptar la tecnología a diferentes niveles de público, y un buen ejemplo es el subtitulado en el mundo audiovisual.

El subtitulado es la transcripción de audio a texto, de manera más o menos literal, de manera que pueda servir como alternativa a aquellas personas que presentan dificultades para la escucha: bien porque tengan una discapacidad, bien porque exista demasiado ruido en el momento de la escucha, o bien porque el audio sea en un idioma poco entendible.

En España el 8% de la población mayor de seis años, es decir, más de un millón de personas, tienen una discapacidad auditiva de distinto tipo y grado. De ellas, más del 97% comunica en lengua oral, según la Encuesta sobre Discapacidades, Autonomía personal y situaciones de Dependencia-EDAD [26] del INE (Instituto Nacional de Estadística). El número de personas va en aumento, lo que hace posible el auge, en los últimos años, del uso del subtítulo.

Otra de las razones está en la globalización del lenguaje. Multitud de contenidos nos llegan a diario registrados con distintos idiomas, y otros como vídeos, películas, reportajes y noticias, entre otras, hacen que sea necesario acompañarla de texto para poder entenderlo.

En cuanto a la importancia de los subtítulos, en España todavía no tiene la divulgación que tiene en otros países. Solamente se usa para el doblaje de contenidos de habla extranjera, y para el entendimiento con la comunidad de personas con discapacidad.

A lo largo de este capítulo se describen algunas tecnologías asociadas a la subtitulación de eventos en tiempo real [27].

2.2.1 Tipo de subtítulos

Los subtítulos pueden ser clasificados según diferentes parámetros. A continuación se citan los distintos tipos de subtitulado según diferentes criterios de clasificación:

2.2.1.1 Según el tiempo

-Subtitulado diferido: En el cual no existe simultaneidad. La preparación de los subtítulos es posterior a la grabación del contenido, pero anterior a la emisión del mismo. Es el más usado en la actualidad, ya que, entre otras cosas, no es tan caro como el que se produce a tiempo real, y la tecnología a implantar es mucho más simple.

Un buen ejemplo de ello es la emisión de series, documentales, películas y proyecciones en los cines, anuncios audiovisuales, etc.

- Subtitulado en tiempo real: A diferencia del anterior, la simultaneidad se tiene que producir en su totalidad, o casi en su totalidad. Es decir, crear los subtítulos y sobreponerlos se tiene que producir a la vez que se produce la emisión del contenido audiovisual. La tecnología a usar es más compleja, y es aquí donde aparece un problema muy común en este tipo de emisiones, que son los retardos.

Ejemplos de este tipo de transcripciones son las retransmisiones deportivas, los programas en directo, tertulias y debates, o cualquier evento en directo.

2.2.1.2 Según el idioma

- Subtitulado intralingüístico: El idioma de los subtítulos es el mismo que el del audio original.

- Subtitulado interlingüístico: El idioma de los subtítulos es diferente al del audio original.

2.2.1.3 Según la funcionalidad

- Subtitulado narrativo: Corresponde a la transcripción del diálogo que se está produciendo.

- Subtitulado informativo: suele añadir información al usuario, sin que ésta forme parte de la obra principal.

- Subtitulado de contenido: suele añadir información al usuario, pero forma parte de la obra principal (y en ocasiones substituye a la misma).

- Subtítulo forzado: es aquel que solo transcribe aquellos tramos que, por idioma u otra causa, necesitan del texto. Es de los subtítulos más populares ya que, con el auge de Internet, muchas de las series y películas presentan este formato.

- Subtítulo cerrado: es el usado cuando se quiere expresar sonidos no verbales, ruidos u onomatopeyas que aporten información al usuario.

2.2.1.4 Según su distribución

- Subtitulado incrustado: está insertado en la propia imagen, por lo que no se puede editar o eliminar. Su principal ventaja es que no requiere de elementos externos para su reproducción.

- Subtitulado flotante: se presenta de manera separada al flujo de video pero ya con formato de imagen. Su principal inconveniente es que se necesita la compatibilidad entre el formato y el reproductor.

- Subtitulado cerrado o aislado: se envía por un flujo alternativo, pero sin renderizar. Tienen menos peso que los casos anteriores, y son más flexibles en su visualización.

2.2.2 Formatos

A la vista de los métodos de distribución existentes y con la multitud de opciones en función del sistema, hay una gran variedad de formatos para el subtitulado [28]:

- SubRip (.srt)
- DVB-Sub: Subtítulos para Difusión de Video Digital.
- Universal Subtitle Format (.usf)
- Micro DVD (.sub)
- SubStation Alpha (.ssa) y Advanced SubStation Alpha (.ass)

Todos los anteriores son los más comunes y todos ellos tienen formato de texto plano, excepto el DVB-Sub, que es el único que permite el formato imagen y el texto plano.

2.2.3 Tecnologías de subtitulado en tiempo real.

El método manual de introducción de subtítulos tiene un índice de errores bastante bajo pero es lento y tedioso. Por esta razón, este método no es válido en los sistemas de subtitulado en tiempo real, donde es imprescindible la transcripción inmediata del discurso.

Los principales subsistemas funcionales del subtitulado en tiempo real son la trascripción de voz a texto, la generación de subtítulos y la transmisión/presentación de los subtítulos, tal como se muestra en la siguiente figura:

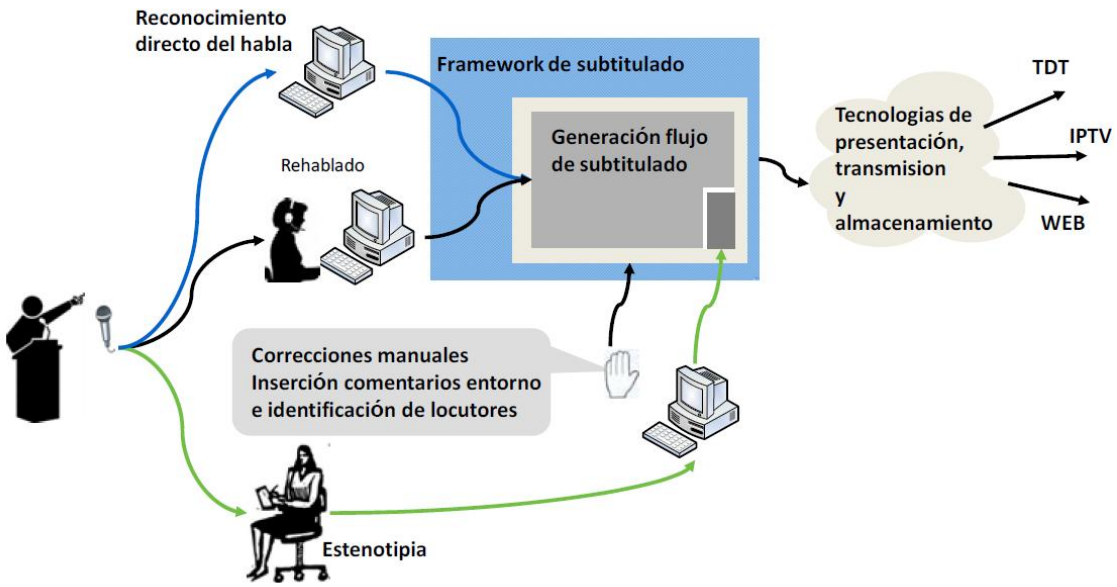

Figura 8. Arquitectura genérica de un sistema de subtitulado en tiempo real.

La transcripción de voz a texto puede llevarse a cabo mediante sistemas de teclado y estenotipia computerizada o con un sistema de reconocimiento automático del habla.

2.2.3.1 Estenotipia computerizada

Es una técnica basada en recoger los signos de puntuación y la información sonora relevante por parte de una persona, la cual está entrenada y capacitada para teclear en una máquina de estenotipia a la misma velocidad del discurso. La máquina de estenotipia permite transcribir sílabas o palabras, mientras que un ordenador asociado genera el subtitulado final.

2.2.3.2 Sistema de reconocimiento automático del habla

"Es el procedimiento por el cual se convierte una señal acústica, capturada por un micrófono, en un conjunto de símbolos de un diccionario dado. Dichos símbolos están generalmente asociados con elementos semánticos de tipo palabra" [29].

Las tecnologías usadas para el ASR permiten la viabilidad del subtitulado masivo de eventos audiovisuales en directo; sin embargo, los mejores sistemas ASR aún están lejos de las prestaciones que pueden dar con respecto a las capacidades del ser humano. Por esta razón, se debe impulsar la investigación en la producción y percepción del habla.

Un sistema ASR tiene una serie de etapas bien definidas como se puede apreciar en la siguiente figura:

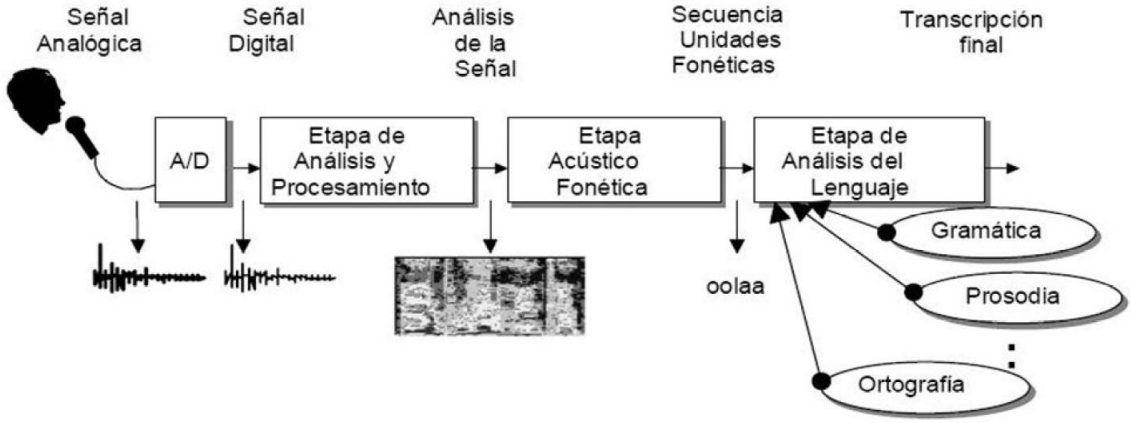

Figura 9. Etapas de un sistema de reconocimiento de habla actual.

a) El procesamiento o análisis del habla, donde habitualmente se hace uso de técnicas STFT (Transformada de Fourier de Tiempo Corto) o POF (Filtrado óptimo probabilístico).

b) La clasificación fonética, que utiliza redes neuronales o HMM (Modelo Oculto Márkov) [30].

c) Análisis en función del modelo del lenguaje, donde se incorpora información prosódica (de acentuación) a un sistema de ASR basado en HMM.

Pero también los sistemas ASR disponen de una serie de inconvenientes como:

- Problemas técnicos relevantes como una gran potencia de cálculo, saturación en el sistema o incompatibilidad con otros programas informáticos.
- No todos los sistemas proporcionan reconocimiento del hablante por lo que pueden producirse retardos serios entre el contenido original y el subtitulado asociado.
- La tasa de acierto en sistemas altamente ruidosos o no entrenados se encuentra siempre por debajo de lo esperado.
- La transcripción obtenida necesita ser corregida, además de asignar las identificaciones de los locutores y añadir información de ciertos efectos sonoros.

El rehablado

Usa también un sistema ASR pero en este caso hay un discurso original que es repetido, de la manera más fiel posible y en unas condiciones óptimas de acústica, por una persona que tiene una formación específica y con un perfil de entrenamiento ya creado. El resultado es que se reduce bastante el número de errores generados en el reconocimiento.

Las principales desventajas son el coste elevado, la necesidad de formación de los "rehabladores" y además se requiere una gran concentración para llevarlo a cabo.

2.3 Sistemas ASR en la actualidad

En la actualidad existen multitud de sistemas ASR, tanto comerciales como gratuitos, capaces de ejecutarse en distintos tipos de plataformas software, desde los sistemas operativos tradicionales de sobremesa como Windows, Linux o Mac OS, hasta los sistemas operativos móviles más populares como Android o iOS.

2.3.1 Sistemas ASR para SO de sobremesa

Sin duda, los dos de los más importantes son Dragon Naturally Speaking de Nuance, que es el reconocedor utilizado en el servidor de APEINTA, y por otro lado, ViaVoice, desarrollado por IBM. A continuación, se detallarán ambos junto con otros que se han considerado de interés también y finalmente se comentará una técnica de optimización para sistemas ASR.

2.3.1.1 Dragon Naturally Speaking

Dragon Naturally Speaking (DNS) [31] es un software de reconocimiento de voz inicialmente desarrollado por *Dragon Systems* de Newton, Massachusetts, que más tarde fue adquirida por *Nuance Communications* [32], anteriormente conocido como *ScanSoft*.

Se ejecuta en sistemas Windows siendo la primera versión lanzada en Junio de 1997 y el paquete más reciente, la versión 13, en Agosto de 2014, la cual soporta las ediciones de 32 bits y 64 bits de Windows 7, 8 y Windows 10. La versión de Mac OS se llama *DragonDictate* o Dragón para Mac [33].

Se trata de una herramienta útil porque posibilita dictar mensajes de correo electrónico, realizar búsquedas o navegar por Internet, y modificar documentos, hojas de cálculo y presentaciones por voz. Por otro lado, presenta ventajas como un entrenamiento previo sencillo, una curva de aprendizaje corta en su uso y soporte para los siguientes idiomas: Inglés, Francés, Alemán, Italiano, Español, Holandés y Japonés.

Una de las características más notables de DNS es su mecanismo de comparación entre tres bases de datos para realizar todas las operaciones básicas: la primera tiene registros de muestras de voz del usuario que DNS ha asociado a diferentes sílabas; la segunda crece conforme se va usando el programa y contiene un vocabulario de usuario, y la tercera consiste en un corpus que contiene un gran número de expresiones que aparecen a lo largo de documentos anteriores que el propio usuario ha ido creando, así como un registro del número de veces que ha dictado determinadas palabras. Durante el dictado de un texto por parte del usuario, DNS hace uso de la primera base de datos para identificar las correspondientes sílabas y, a continuación, averigua las palabras que corresponden a las diferentes secuencias de sílabas identificadas recurriendo a la segunda base de datos. Se puede identificar una palabra o un símbolo (por ejemplo, «coma»), o una orden determinada (por ejemplo, «nueva línea» introduce un salto de línea) y a medida que se realiza este proceso, se van mostrando por pantalla los resultados. En caso de duda entre dos palabras o expresiones fonéticamente similares, DNS escoge la más adecuada consultando al corpus la frecuencia con la que se utiliza una u otra en el contexto del dictado.

Por otro lado, DNS proporciona su propio editor de texto, llamado *Dragonpad*, además de otras herramientas, y prácticamente puede usar cualquier otro programa del entorno Windows con el que habitualmente se pueda escribir texto con un teclado, ya sea un software comercial o uno de código abierto (MS Word, Excel, PowerPoint, Explorer, Bloc de notas...). Además, los perfiles de usuario creados y entrenados son accesibles a través de diferentes equipos en el mismo entorno de red, aunque la configuración de hardware y de audio debe ser idéntica en todos los equipos implicados.

Inicialmente existían tres variantes que se mantuvieron hasta la versión 10 del programa:

- *Standard*: es la más sencilla y únicamente permite operaciones básicas como dictar texto, números y órdenes en varios programas.
- *Preferred*: es la versión intermedia y es capaz de reproducir con voz propia lo que ha dictado el usuario con anterioridad, utilizar una grabación de voz para escribir un dictado y también se posibilita la importación y exportación de archivos de usuario.
- *Professional*: es la versión más avanzada y posibilita además trabajar en red, con macros, y disponer de distintos tipos de vocabularios concretos y específicos para cada usuario.

Sin embargo, DNS ha continuado evolucionando y se mantiene como una herramienta muy extendida para uso particular o empresarial donde sea necesario el reconocimiento de voz, con cinco versiones comerciales para PC disponibles desde la tienda web [34]:

- Dragon 13 Home: 99 €
- Dragon 13 Premium: 169 €.
- Dragon 13 Premium Mobile: 249 €.
- Dragon 13 Premium Wireless: 249 €.
- Dragon 13 Professional: desde 599 €

2.3.1.2 ViaVoice

ViaVoice [35] es un software de ASR desarrollado inicialmente por IBM (*International Business Machines Corporation*) [36]. La última versión estable es la 10.5, lanzada en Junio de 2005, y está diseñada principalmente para su uso en dispositivos embebidos con Windows o Mac OS.

En 1997 ViaVoice fue presentado por primera vez de forma pública, siendo además el primer software de dictado de voz con soporte a chino y japonés y con la tecnología de reconocimiento de habla continua de gran vocabulario (LVCSR). Dos años más tarde, IBM lanzó una versión gratuita del programa y en 2003, IBM vende a ScanSoft, propietaria del producto DNS, los derechos exclusivos de distribución mundial de ViaVoice.

Para mejorar la precisión del reconocedor de voz se puede realizar un entrenamiento con textos específicos con unos pocos cientos de frases. Los datos registrados se utilizan para ajustar el modelo acústico para cada usuario específico.

ViaVoice ha evolucionado hacia un programa con aplicaciones para grandes empresas y su uso en red. Algunas de las versiones disponibles son: Edición Avanzada, Edición Estándar, Edición Personal, Edición para Mac OS X y Dictado simple para Mac.

2.3.1.3 Media Mining Indexer

Media Mining Indexer [37] (MMI) es una solución software propiedad de *SAIL LABS* (Speech, Artificial Intelligence and Language Labs) [38] con soporte para Windows y Linux.

Puede procesar la voz desde múltiples fuentes en varios formatos y producir en tiempo real la salida en forma de texto incluso con altos niveles de ruido de fondo. La salida en XML (lenguaje de marcas extensible) facilita la carga en diversos sistemas de gestión de ficheros.

El proceso ASR se realiza en una secuencia de pasos: primero procesa el audio entrante, luego los segmentos de audio se clasifican según contengan muestras de voz o no y, a continuación, se aplica el reconocimiento de voz a los segmentos identificados como hablados. Todas las tareas se realizan en tiempo real para grandes vocabularios (más de 64.000 muestras).

El motor de reconocimiento de voz es independiente del lenguaje y funciona para una gran variedad de idiomas como: Inglés, francés, español, alemán, árabe, ruso, noruego o polaco.

La arquitectura es escalable y soporta la configuración de múltiples máquinas, proporcionando una solución rentable para estaciones de televisión pequeñas y medianas por ejemplo.

2.3.1.4 DynaSpeak

DynaSpeak [39] es un motor de reconocimiento de voz de alta precisión y de tamaño reducido propiedad de la empresa norteamericana *SRI International* [40], que puede incluirse en sistemas embebidos o en sistemas de gran escala de la industria y productos militares.

Su motor puede ser adaptado a una gran variedad de procesadores y sistemas operativos al disponer de un SDK en C/C++ y Java ganando flexibilidad en el diseño del producto. Soporta Windows, Mac OS X y Linux posibilitando el desarrollo de versión cliente y servidor.

Debido a que DynaSpeak ha sido desarrollado para aplicaciones embebidas de campo, incorpora técnicas patentadas de SRI que aumentan el rendimiento del reconocimiento mediante la adaptación del dictado a cada usuario, adaptación al micrófono, fin de la detección de voz, reconocimiento de voz distribuido y la robustez frente al ruido. El propio reconocimiento de la voz se basa en técnicas de HMM.

2.3.1.5 LumenVox

LumenVox [41] ASR es una solución de software para reconocimiento de voz creada y soportada por la empresa del mismo nombre. Dispone de versiones de 32 y 64 bits para Linux y Windows, proporcionando soluciones para el Entendimiento de Lenguaje Natural (NLU) a través del desarrollo de Modelos de Leguaje Estadísticos (SLM).

El funcionamiento es muy sencillo: inicialmente existe una serie de registros vocales de una lista de frases textuales, llamados en su conjunto gramática. La gramática es usada para restringir la búsqueda mediante comparaciones forzando al ASR a elegir la mejor opción.

Se integra con más de 25 plataformas de voz al disponer de una API propietaria y una solución estandarizada como el protocolo de control de recursos de media (MRCP).

2.3.1.6 Verbio

Verbio ASR es el motor de reconocimiento del habla propietario de Verbio Technologies SL [42], una empresa especializada en el desarrollo de tecnologías del habla con sede en Barcelona (España). Se encuentra disponible en los distintos idiomas utilizados en España, Portugal y en muchos países latinoamericanos, incluyendo también una versión en inglés.

El ASR se presenta en una versión telefónica así como una versión para micrófono. Además existe un SDK con una potente API en la nube que permite una integración fácil y sencilla.

Verbio ASR hace uso de LVCSR y lo combina con la comprensión del lenguaje natural para hacer soluciones automatizadas inteligentes: enrutamiento de llamadas, sistemas de respuesta de voz interactiva y transcripciones automáticas.

Por último, este motor está enfocado principalmente para trabajar en entornos telefónicos aunque tras un proceso de adaptación, puede ser compatible en otros recursos o modelos acústicos, como por ejemplo cualquier entorno que disponga de un sistema de manos libres, en automatización de centros de llamadas o en el campo de la Domótica.

2.3.1.7 CMU Sphinx

CMU Sphinxes [43] es un grupo de sistemas de reconocimiento de voz desarrollado en la Universidad Carnegie Mellon de Pittsburgh (Estados Unidos). Se trata de una serie de reconocedores de voz (Sphinxes 2-4) y un entrenador de modelo acústico llamado *SphinxTrain*.

En el año 2000 el grupo de desarrolladores de Carnegie Mellon decidieron publicar el código fuente de Sphinxes 2 y un año más tarde de Sphinxes 3 bajo una licencia BSD (Distribución de software berkeley). La última versión está escrita en Java y se publicó en Agosto de 2015.

Los reconocedores de voz se proporcionan con modelos acústicos predefinidos, aplicaciones de ejemplo, software para el entrenamiento del modelo acústico, compilación del modelo de lenguaje y un diccionario de pronunciación de dominio público, *CMUdict*.

2.3.1.8 Julius

Julius [44] es un software libre para reconocimiento de voz publicado con licencia BSD y diseñado para plataformas GNU y UNIX aunque también hay versiones para Windows. Está basado en trigramas y en modelos HMM y puede transcribir a texto en tiempo real.

Julius se originó en 1977 a partir de un kit de software libre para investigación en técnicas de LVCSR y el trabajo ha sido continuado por el Consorcio de Reconocimiento de habla continua en Japón de 2000 a 2003. La versión más reciente de Julius es la 4.3.1, disponible desde el 15 de Enero de 2014.

Para trabajar con Julius es necesario un modelo de lenguaje y un modelo acústico para el lenguaje en concreto. Se han distribuido modelos en japonés e inglés para utilizarlos con el motor de reconocimiento de voz de Julius, además el proyecto VoxForge [45] trabaja en la creación de nuevos modelos acústicos en varios idiomas.

2.3.1.9 HTK

HTK (modelo oculto de Markov Toolkit) [46] es un conjunto de herramientas y bibliotecas disponibles en C para la construcción, evaluación y modificación de modelos HMM.

Para estimar los parámetros de modelos HMM se incluyen dos herramientas para la inicialización y otras dos para la reestimación de los mismos. Además es capaz de distinguir dos clases de modelos HMM: modelo de palabras completas y modelo de parte de palabra (habitualmente fonemas).

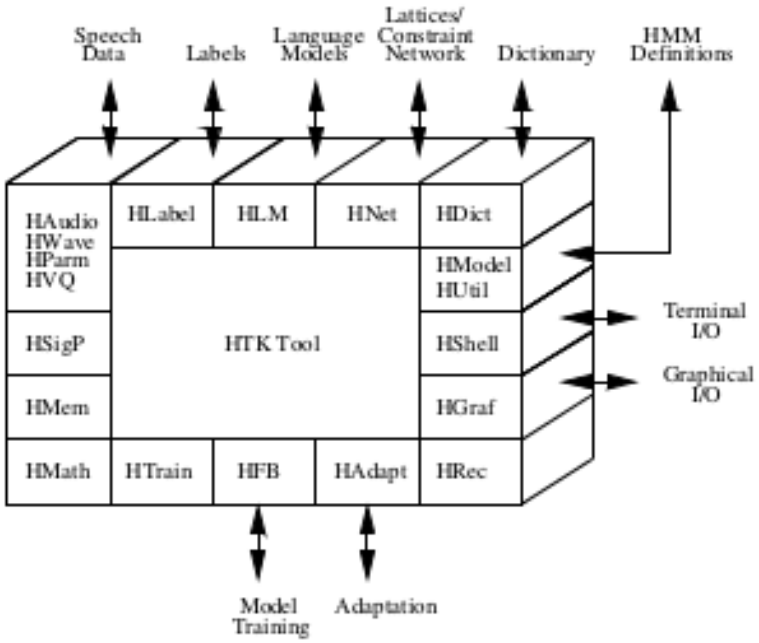

Figura 10. **Arquitectura Software de HTK.**

Su uso principal es la investigación de reconocimiento de voz aunque se ha utilizado también para la investigación de síntesis de voz, reconocimiento de caracteres y secuenciación de ADN.

Fue originalmente desarrollado por el Departamento de Ingeniería de la Universidad de Cambridge (CUED). En 1993 *Entropic* adquirió los derechos para distribuir HTK y su desarrollo fue transferido plenamente a este laboratorio dos años más tarde. En 1999 Microsoft compró Entropic y los últimos años Microsoft ha derivado de nuevo la responsabilidad en la distribución, soporte y desarrollo de HTK a CUED a través del sitio web htk3.

Aunque Microsoft mantiene el derecho de autor del código original de HTK, cualquier desarrollador puede realizar cambios en el código fuente e incluir los aportes en la web htk3.

Por último, HTK está disponible como una distribución de código fuente en C, por lo que se necesita una herramienta de compilación para su instalación en plataformas Windows y Linux.

En caso de dudas acerca de la instalación o manejo de de las herramientas que componen HTK se dispone de una gran documentación en el HTKBook.

2.3.2 Sistemas ASR para SO móviles

Una de las características de los dispositivos móviles en las que se ha avanzado mucho es en el reconocimiento de voz y es que las tabletas y los smartphones no cuentan generalmente con teclados físicos, por lo que el desarrollo de métodos de entrada como el dictado se convierte en algo fundamental a la hora de simplificar las tareas.

Aparte de *Google Now*, *Siri y Cortana*, numerosas aplicaciones móviles gratuitas o de pago utilizan el reconocimiento de voz para responder a diferentes necesidades corrientes o profesionales. A continuación se introduce una pequeña selección de este tipo de aplicaciones.

2.3.2.1 Google Now

Google Now [47] es un asistente personal incluido dentro de la aplicación *Google Search* para móviles Android de forma gratuita. También está disponible para iOS desde Abril de 2013.

El 9 de julio de 2012 Google Now fue incluido por primera vez en Android 4.1 y un año más tarde, *Popular Science* nombró a la herramienta como la "Innovación del Año 2012".

Utiliza una interfaz de usuario de lenguaje natural para responder preguntas realizadas por el usuario mediante comandos de voz, recomendar información o realizar acciones a través de un conjunto de servicios web. En función de los hábitos de búsqueda del usuario, Google ofrece también de forma pasiva una predicción de la información deseada por el usuario. También aprovecha funcionalidades del proyecto *Knowledge Graph* de Google, un sistema para ensamblar resultados de búsquedas muy concretas mediante el análisis de su significado.

2.3.2.2 Siri

Siri [48] es una aplicación nativa y gratuita disponible para iOS y propietaria de Apple con funciones de asistente personal para procesar el lenguaje natural del usuario.

La herramienta fue creada en Diciembre de 2007 en el Centro de Inteligencia Artificial de SRI International, dentro del proyecto CALO, financiado por DARPA (Agencia de Proyectos de Investigación Avanzados de Defensa). Finalmente, Apple adquirió Siri el 28 de Abril de 2010.

La integración con iOS es total permitiendo la interacción por comandos de voz con otras aplicaciones. La tecnología de Nuance es también la base del ASR que utiliza Siri.

2.3.2.3 Cortana

Cortana [49] es el asistente personal desarrollado por Microsoft para todos los dispositivos con Windows Phone desde la 8.1 y para Windows 10. Se esperan versiones para iOS y Android.

Fue lanzada en primer lugar en beta en los Estados Unidos a mediados de 2014 y después también en Reino Unido y China. A finales de 2014 estaba disponible en el resto de países.

El procesamiento de lenguaje natural de Cortana se ayuda de una base de datos de búsqueda semántica con una tecnología que heredo Microsoft al comprar *Tellme Networks* en 2007.

Cortana se ha adaptado a los patrones cotidianos y culturales de cada región e idioma porque es diseñado de forma específica para cada país. Utiliza *Bing, Yelp y Foursquare* como bases de datos y a partir de Windows Phone 8.1 sustituye a la búsqueda integrada de Bing.

2.3.2.4 Hound

Hound [50] es una aplicación de asistencia actualmente solo disponible para dispositivos Android en los Estados Unidos. Fue creada por la empresa californiana *SoundHound*.

Hound une el reconocimiento de voz y el procesamiento del lenguaje en un único motor de búsqueda en lugar de realizarlo en tareas distintas y separadas como Siri o Google Now.

2.3.2.5 S Voice

S Voice [51] es un servicio gratuito de reconocimiento de voz desarrollado por Samsung que se incorporó por primera vez en el Galaxy S III en Mayo de 2012.

Proporciona comandos de voz para acelerar algunas operaciones y el servicio sólo funciona con una conexión de red, lo que parece indicar que el procesamiento se realiza en la nube.

2.3.2.6 Sirius

Sirius [52] es un proyecto gratuito y de código libre que se ha originado en la Universidad de Michigan. Consiste en un asistente personal para reconocimiento de voz y de imagen, uso de lenguaje natural y sistema de preguntas y respuestas. Sirius recibe las órdenes en el smartphone, procesa la información en la nube y genera la respuesta de voz.

El sistema ASR incluye aprendizaje mediante inteligencia artificial, de forma que los datos se interpretan de forma distinta con el tiempo.

2.3.2.7 Soluciones Dragon

Actualmente Nuance presenta 5 soluciones [53] relacionadas con el ASR en smartphones:

Dragon Remote Microphone: Es una aplicación para iOS y Android que conecta el dispositivo de forma inalámbrica a un PC o MAC con Dragon instalado y realiza el dictado en tiempo real.

Dragon Recorder: Es una aplicación para iOS que graba audio. El fichero de audio se transcribe a texto posteriormente por un producto Dragon en un PC o MAC.

Dragon Dictation: Es una aplicación para iOS que transcribe a texto la voz en tiempo real y el resultado se usa para elaborar mensajes de correo electrónico, entradas de blog etc.

Dragon Mobile Assistant: Es una aplicación para iOS y Android que realiza las funciones de asistente personal inteligente como podría hacerlo Google Now o Siri.

Dragon Anywhere: Es una aplicación para Android o iOS diseñada para profesionales y basada en la nube. Está anunciada para otoño de 2015 y será necesaria una subscripción.

2.4 Subtitulado en tiempo real en entornos educativos o con tecnologías móviles

Para las personas sordas o con problemas de audición puede resultar un desafío comunicarse con otras personas a través del lenguaje hablado y más en una sociedad donde representan una minoría. Se puede observar claramente en lugares públicos como los centros de enseñanza y universidades, donde puede no haber formas accesibles de comunicación para esta clase de personas. De hecho, esta barrera es especialmente importante a la hora de establecer una comunicación entre el profesor y el alumno con discapacidad auditiva.

En esta sección se pretende mostrar y evaluar algunos escenarios y herramientas para la transcripción de sonido a subtítulos especialmente en entornos educativos o a través de tecnologías móviles. En concreto, algunos escenarios que se van a citar continuación son muy similares al que se pretende conseguir con este proyecto y su impacto en APEINTA.

2.4.1 Ai-Live

Ai-Live [54] fue desarrollado originalmente para proporcionar acceso a personas con discapacidad auditiva en reuniones de trabajo, clases, conferencias o cualquier evento similar. También es utilizado por personas con parálisis cerebral, dificultades de aprendizaje, preferencias con el aprendizaje visual y dificultades para el aprendizaje del inglés.

Para lograr el más alto estándar de calidad, Ai-Live utiliza sistemas de subtítulado y taquígrafos altamente capacitados y entrenados que escuchan el flujo de audio en directo y repiten todo lo que escuchan, incluyendo puntuación y gramática, para que los sistemas ASR lo conviertan a texto y se pueda transmitir al dispositivo de destino en directo (ver figura 11).

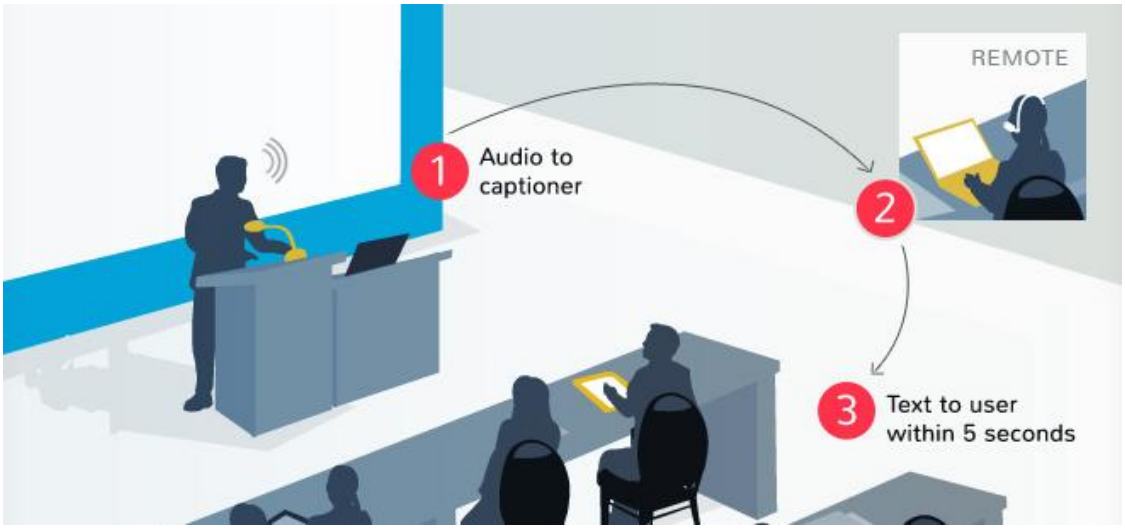

Figura 11. Sistema Ai-Live para un evento en directo.

Las palabras habladas aparecen en la pantalla de los dispositivos segundos después de que se articulan. Incluso el sistema es capaz de escuchar la lección, resumir el contenido y entregar los resúmenes como subtítulos.

Algunas opciones y características ofrecidas por Ai-Live son:

- Seleccionar las fuentes y tamaños del texto.
- Elegir una paleta de colores para mejorar la velocidad de lectura en un 400%.
- Chatear directamente con un transcriptor mediante el panel de chat Ai-Live para confirmar la configuración de audio y otras opciones.
- Agregar palabras a un diccionario personal mediante avisos al transcriptor.
- Almacenar las transcripciones de las sesiones para su posterior búsqueda y consulta.

2.4.2 Hamilton CapTel

Hamilton CapTel [55] para smartphones permite transformar la voz durante las conversaciones telefónicas en subtítulos de forma casi instantánea. De manera similar al subtitulado en televisión, las transcripciones de una conversación se muestran en la pantalla del smartphone. Hay versiones disponibles para dispositivos Android e iPhone y el servicio está disponible las 24 horas del día sin coste adicional.

Los requisitos para hacer uso de esta aplicación y servicio son:

- Un smartphone y redes inalámbricas compatibles con el servicio.
- Auricular con manos libres que pueda funcionar con audífonos o un implante coclear.
- Registro de una cuenta de Hamilton CapTel para realizar y recibir llamadas.
- Plan de voz y datos en el dispositivo móvil.

Si se desea usar el servicio de subtitulado para una llamada, es necesario realizar la misma desde la aplicación Hamilton CapTel con el auricular conectado correctamente y listo para usar. Previamente a realizar la llamada de voz real, la aplicación se conecta con el servicio de subtitulado a través de un centro de llamadas propio de este servicio y finalmente ya se continua con el proceso de la llamada habitual a través de la operadora contratada.

2.4.3 E-Scribe

E-Scribe [56] es un sistema online basado en un prototipo web para la transcripción de voz en tiempo real para personas con problemas de audición, técnicamente basado en telefonía IP y visualizando la transcripción en directo en una página web.

El proyecto se ha desarrollado en el centro de aplicaciones móviles de la Universidad Técnica de Praga y es patrocinado por la Fundación Vodafone. Ahora está implementado con éxito en conferencias para estudiantes con discapacidad auditiva de Universidad Técnica de Praga.

La voz de los usuarios se transmite a través de VoIP o por un servicio de telefonía conmutada al centro de la transcripción en directo (ver figura 12). La voz se transcribe por mecanografía rápida usando MS Word, que está disponible a través del formulario web en el servidor, y más tarde combinando sistemas ASR con correctores de errores humanos para automatizar el proceso de transcripción. Por último, el discurso transcrito se proyecta en una pantalla a través de un navegador web conectado al servidor. Sin embargo, la transcripción está disponible también desde cualquier lugar con conexión a Internet.

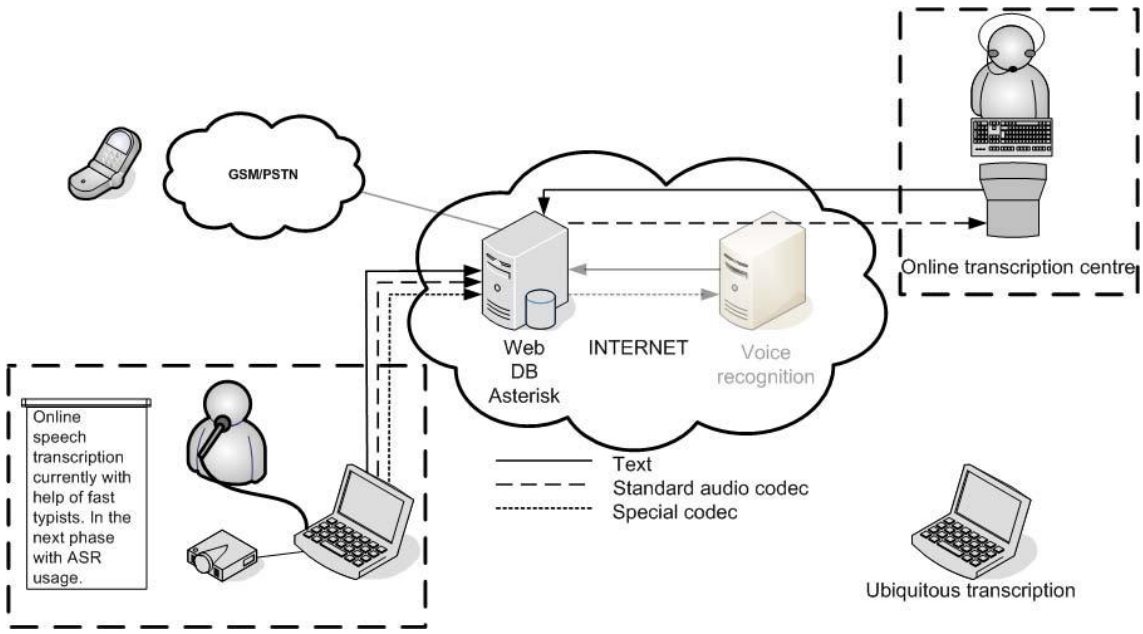

Figura 12. Sistema e-Scribe

El acceso al sistema es posible desde teléfonos normales, teléfonos móviles pero también con teléfonos SIP con clientes de software.

2.4.4 Liberated Learning

El *Liberated Learning Consortium* [57] es un grupo de investigación internacional dedicado a mejorar el acceso a la información por medio del reconocimiento de voz basado en sistemas de subtítulado y transcripción.

Este consorcio inició en 2009 diálogos con diversas organizaciones canadienses relacionadas con la discapacidad para promover su nuevo proyecto *Liberated Learning* [58]. Durante este proyecto de investigación de dos años, desde Enero 2010 hasta Abril de 2012, los estudiantes con discapacidades utilizaron un ASR con un Sistema de Transcripción Alojado (HTS) para acceder a la información en directo.

Los objetivos del proyecto eran:

- Incrementar el uso de tecnologías de reconocimiento del habla para servicios de transcripción o de subtitulado.
- Incrementar el acceso individual a las tecnologías de reconocimiento del habla para servicios de transcripción o de subtitulado.
- Desarrollo de estrategias para permitir a otras organizaciones relacionadas con la discapacidad a utilizar eficazmente las tecnologías de reconocimiento del habla.

En cuanto al sistema ASR usado, HTS es un sistema que transcribe automáticamente audio o vídeo en transcripciones multimedia accesibles. Mientras que las transcripciones tradicionales se generan normalmente escuchando y escribiendo manualmente lo que se escucha, las transcripciones multimedia provienen de un texto transcrito que se sincroniza con una fuente de lenguaje hablado.

Además, el proyecto inició el desarrollo de dos nuevas plataformas HTS:

- IBM introdujo *iTrans* que aprovecha la nube de IBM para la transcripción de voz.
- Nuance también contribuyó con un kit de desarrollo de software para su reconocido motor de voz Dragon. También incluye opciones para una amplia variedad de idiomas, incluyendo el francés, que es un factor crítico de éxito en curso para la región.

Por último, los investigadores calcularon estadísticas de uso y rendimiento técnico para el HTS, incluyendo el análisis de las grabaciones subidas, los metadatos asociados, y una WER (Tasa de error de palabra) para el reconocimiento de voz, dando como resultado que un tercio de todas las transcripciones tenían un 15% de error o baja WER (85% precisión).

2.4.5 ParaKeet

ParaKeet [59] es un sistema para generar de forma eficiente texto tras el reconocimiento de la voz en dispositivos de pantalla táctil y usando un micrófono Bluetooth.

Un caso de uso concreto es el reconocimiento de voz en el dispositivo móvil Nokia N800 con este sistema. Se utiliza un micrófono Bluetooth para la captura de audio y el reconocedor *PocketSphinx* para el reconocimiento de voz. Usando un vocabulario de 5000 palabras se ha demostrado que se puede lograr una buena precisión y rendimiento en el reconocimiento casi en tiempo real. La interfaz de corrección se basa en la visualización de las posibles alternativas para cada palabra suponiendo la mejor hipótesis.

2.4.6 Scribe4Me

Scribe4Me [60] era una herramienta de transcripción de voz para dispositivos móviles orientada a personas con problemas de audición. La herramienta es configurable y puede capturar datos multimedia. Existe una versión escrita en *Java Micro Edition* y otra en C #.

El sistema funciona con una arquitectura cliente-servidor de forma que se ha diseñado también un servidor de transcripción de voz a texto escrito en Java, cuyo interfaz es manejada por un transcriptor. El mecanismo de funcionamiento es el siguiente:

- Inicialmente, el transcriptor utiliza el servidor para agregar participantes a la implementación y realizar la configuración de los mismos. Cada participante por su parte dispone de su propio dispositivo móvil con Scribe4Me que debe registrar en el servidor. El transcriptor supervisa las solicitudes entrantes en una línea de tiempo.
- Los participantes pueden solicitar en cualquier momento una transcripción de los últimos 30 segundos de audio pulsando un botón en la interfaz de Scribe4Me. La aplicación almacena los datos de audio en local e intenta enviarlo el flujo al servidor.
- La solicitud se recibe y se presenta en una línea de tiempos en el servidor. A partir de ese momento el transcriptor escucha el archivo de audio adjunto y responde enviando una transcripción. La plataforma soporta una gran variedad de métodos para enviar datos a los participantes. Lo más habitual es que el participante reciba la transcripción como un mensaje de texto.

2.5 Entorno socio-económico

El entorno en el que se desenvuelve el sector audiovisual en la actualidad está marcado por cuatro problemas básicos:

- No se han ido actualizando las pautas básicas de legislación en este sector, con lo que denota una obsolescencia notoria en contraposición al mercado.
- La multitud de medidas que se han establecido dentro del marco de regulación han hecho que la información que subyace sea cada vez más confusa y compleja del sector.
- La poca verosimilitud entre el marco normativo y la realidad, hace que cada vez se hagan más insostenibles las estructuras financieras y, además, aparezcan nichos de mercado en los cuales no se ampara la ley.
- Está inmerso en un macro sector, lo que hace que pueda ser arrastrado hacia un punto de donde no pueda salir fácilmente, con lo que supondría perder oportunidades en el mercado y una posible evolución.

Por otra parte, el desarrollo socio-económico, el apoyo institucional, la normativa autonómica, la creciente dimensión empresarial, el desarrollo tecnológico, entre otros factores, han contribuido a potenciar el crecimiento del sector.

Para conseguir mitigar estos errores, tanto los agentes políticos y reguladores como las empresas están aunando esfuerzos para conseguir encauzar y poner fin al caos que se ha ido produciendo con el paso de los años.

2.6 Marco regulador

En los últimos años se han producido varios cambios establecidos por el Gobierno, como consecuencia del auge de nuevas plataformas tecnológicas, y teniendo que intervenir otros organismos como la Comisión Nacional de los Mercados y de la Competencia, para regular posibles ilegalidades y prácticas abusivas.

Casi el total de las competencias son asumidas por el Ministerio de Industria, Turismo y Comercio, y en concreto por la Secretaría de Estado de Telecomunicaciones y para la Sociedad de la Información (SETSI). Ésta se encarga de los aspectos más esenciales e importantes relacionados con el sector audiovisual.

Surgió una nueva norma 153010:2012, acerca del subtitulado para personas sordas y personas con discapacidad auditiva. Anula y sustituye a la Norma UNE 153010:2003, y está elaborada por el Comité Técnico de Normalización (CTN) 153 de AENOR [61].

Tiene como objetivo, entre otros, generar unos requisitos mínimos de calidad y trazar unas pautas de homogeneidad para el subtitulado de aquellas personas con este tipo de discapacidad. Nació con el fin de adaptarse a nuevas tecnologías, como los DVD, la televisión digital o la televisión emitida a través de Internet.

En su resolución se distinguen entre aspectos temporales, visuales, de identificación de locutores y criterios de división de texto a cumplir.

2.7 Discusión

El propósito de esta sección es justificar la elección de las diferentes soluciones descritas a lo largo del Estado del Arte que han resultado más apropiados para conseguir la viabilidad de este proyecto. Para ello, se realiza un estudio comparativo entre la solución elegida y las alternativas descritas en los siguientes ámbitos: sistema operativo móvil de la aplicación desarrollada, el sistema ASR para la conversión a texto y el sistema de subtitulado en tiempo real en entornos educativos o con tecnologías móviles.

Sistema operativo móvil

El sistema operativo móvil de la aplicación desarrollada en este proyecto fin de carrera debe cumplir en la medida de lo posible los siguientes requerimientos: disponibilidad de soluciones de código abierto para el ámbito del proyecto, expectativas de corta duración para los desarrollos necesarios y una buena cuota de mercado en términos de dispositivos o usuarios.

En el caso de Blackberry OS, el nivel técnico inicial para comenzar a desarrollar aplicaciones es relativamente alto, lo que aumentaría los costes de desarrollo. Además, la cuota de mercado para esta plataforma es de las más bajas. Por estas razones, Blackberry OS ha sido rechazada.

Las soluciones para Windows 10 han sido rechazadas también a causa de la falta de aceptación general de su versión predecesora, Windows Phone, siendo el número de aplicaciones para esta plataforma aún pequeña. Por último, no es una plataforma de código abierto.

Por último, en el mismo caso que Windows Phone, iOS ha sido rechazada porque no es una plataforma de código abierto. Por otra parte, el lenguaje de programación y las herramientas para el desarrollo son propietarios de Apple, por lo que también podría suponer un período de desarrollo más largo. Además, cualquier desarrollo para iOS debe pasar las restricciones de Apple para ser publicado.

En base a estas decisiones, Android es la alternativa más viable para este proyecto. A continuación, se exponen las principales razones:

- Es un sistema operativo apoyado por muchas empresas importantes.
- Es una plataforma de código abierto por lo que la comunidad de desarrolladores es grande ofreciendo un gran apoyo para los nuevos desarrolladores.
- Desarrollar en Android es relativamente fácil ya que se necesitan sólo conocimientos de Java y el manejo de un IDE tan genérico como Eclipse.
- Posee actualmente la mayor cuota de mercado en número de usuarios y aplicaciones.
- Cuenta con una gran cantidad de proyectos de código abierto.

Sistema ASR

El software ASR que usa el servicio de transcripción en tiempo real de APEINTA es una versión de escritorio para Windows de DNS. Por tanto, no ha sido necesario realizar pruebas con otras alternativas de sistemas ASR pues el CESyA contaba con una licencia para DNS y se trata en este caso de un requisito impuesto por el proyecto APEINTA.

En cualquier caso, a continuación se compara DNS con las otras alternativas descritas:

La mayoría de los productos ASR descritos para sistemas operativos de sobremesa (ver tabla 2) tienen soporte para varios sistemas operativos, excepto *Verbio* que solo está disponible para Windows, permiten el reconocimiento de distintos idiomas y posibilitan integrar el módulo ASR con ciertos sistemas mediante un SDK. Además, la mayoría usan HMM.

Sistema ASR	Propietario Fundador	Plataformas soportadas	Licencia	Soporte idiomas	SDK/APIs disponibles	Tecnología usada
DNS	Nuance	Windows y Mac	Propietaria	Si	Si	HMM
ViaVoice	IBM	Windows y Mac os	Propietaria	Si	-	HMM
MMI	SAIL LABS	Windows y Linux	Propietaria	Si	Si	-
DynaSpeak	SRI International	Windows, Mac y Linux	Propietaria	Si	Si	HMM
LumenVox	LumenVox	Linux y Windows	Propietaria	Si	Si	HMM
Verbio	Verbio Technologies	Windows	Propietaria	Si	Si	LVCSR
CMU Sphinx	Universidad Carnegie Mellon	Windows, Mac y Linux	BSD	Si	Si	LVCSR
Julius	-	Windows y Linux	BSD	Si	Si	HMM
HTK	CUED, Microsoft	Windows y Linux	Propietaria	Si	Si	HMM

Tabla 2. Comparativa entre sistemas ASR para SO de sobremesa

Aunque *CMU Sphinx* y *Julius* son herramientas gratuitas, quizás la mejor alternativa a DNS sea HTK, por la cantidad de utilidades disponibles y por la gran comunidad de desarrolladores.

La tabla 3 muestra sistemas ASR para plataformas móviles, siendo la mayoría asistentes personales para dispositivos. En cualquier caso, la versión DNS elegida funciona en un servidor.

Sistema ASR	Desarrollador	SO móviles soportados	Gratuito	Soporte idiomas	Tecnología usada
Google Now	Google	Android y iOS	Si	Si	-
Siri	Apple	iOS	Si	Si	HMM
Cortana	Microsoft	Windows Phone, Android e iOS	Si	Si	Búsqueda semántica
Hound	SoundHound	Android	No	Inglés	NLU
S Voice	Samsung	Android	No	Si	ASR en nube
Sirius	Universidad de Michigan	Multiplataforma	Si	Si	HMM, ASR en nube
Soluciones Dragon	Nuance	Android, iOS	No	Si	HMM

Tabla 3. Comparativa entre sistemas ASR para SO móviles

Sistema de subtitulado en tiempo real

El sistema de subtitulado en tiempo real que se describe en este proyecto es muy similar al sistema actual de APEINTA con la salvedad de que se usa un smartphone para grabar y enviar la voz del orador en lugar de usar un micrófono. Por tanto, el motor de reconocimiento del habla así como los posibles retardos por procesado en la transcripción a texto, son muy similares a los ya experimentados con APEINTA.

A continuación se presentan las desventajas de algunos de los sistemas de subtitulado en tiempo real en entornos educativos o con tecnologías móviles que se han descrito con anterioridad en este documento en relación al propuesto en este proyecto, justificando así la necesidad del mismo:

Ai-Live:

Es necesario el uso taquígrafos que repiten todo lo que el orador habla para que los sistemas ASR lo conviertan a texto, lo que conlleva a una mayor complejidad del sistema en términos computacionales. Además, existe un transcriptor personal para que los alumnos puedan personalizar y mejorar en tiempo real los resultados de las mismas transcripciones, pero también supone un gasto adicional de recursos humanos y económicos.

Además, no usa alternativas con tecnologías móviles para la grabación y envío de voz, limitando mucho la capacidad de movilidad del orador.

E-Scribe:

La voz se transcribe por mecanografía rápida usando un software adicional además de un sistema ASR con correctores de errores humanos, por lo que el proceso de automatización de la transcripción resulta más lento.

Por último, el discurso transcrito se proyecta en una pantalla, en lugar de hacerlo de forma individualizada para cada alumno a través de un dispositivo personal.

Liberated Learning:

Una vez que se obtiene el texto transcrito, es necesario sincronizar el mismo con una fuente de lenguaje hablado, por lo que el proceso en general resulta más costoso computacionalmente.

Por otro lado, es necesario el uso de sistemas ASR en la nube, por lo que los requisitos de ancho de banda se multiplican.

Scribe4Me:

Los alumnos solo pueden solicitar una transcripción de los últimos 30 segundos de audio con este sistema.

Además, no existen procesos de automatización de las tareas y hay mucha demora en la recepción de resultados, pues el transcriptor es una persona física que escucha el archivo de audio adjunto y responde enviando una transcripción.

Capítulo 3. Marco de Trabajo

El marco de trabajo en el que se sustenta este proyecto fin de carrera consiste en dos proyectos totalmente independientes entre sí pero que han servido de referencia para la concepción y realización del mismo.

De este modo, se presenta por un lado el proyecto SipDroid y, en segundo lugar, la descripción y evaluación del servicio de subtitulado automático APEINTA.

3.1 SipDroid

SipDroid [62] es una aplicación VoIP para Android que hace uso de SIP (Protocolo de Inicio de Sesión) para realizar llamadas por Internet desde un proveedor VoIP al que se esté subscrito. Es un software libre de código abierto liberado bajo la GNU GPL (*General Public License*).

El proyecto se inició el 12 de marzo de 2009 y alcanzó la versión 1.0 el 12 de julio del mismo año. Las versiones principales que se han ido lanzando son: 1.5 (29 de mayo de 2010) para mejorar la calidad de video, 2.0 (18 de noviembre de 2010) con la posibilidad de conectar con una cuenta de Google Voice, 2.2 (25 de marzo de 2011) para el envío de SMS multimedia, 2.7 (21 de mayo de 2012) para mejorar la latencia, y 3.0 (22 de abril de 2013) con soporte para TLS (Seguridad de la capa de transporte).

En 2010-2011 ganó popularidad en parte debido a su interacción con el servicio Google Voice de Google, permitiendo hacer llamadas haciendo uso de la red de datos. Sin embargo, Google Voice eliminó la posibilidad de conectarse a través de SIP a partir del 08 de marzo de 2011, eliminando también esta funcionalidad en SipDroid.

Hasta el 24 de Julio de 2015 SipDroid ha sido descargada 1.000.000-5.000.000 veces desde Google Play y la última versión es la 3.7.

Algunas características a destacar son:

- Se pueden utilizar dos cuentas SIP simultáneamente.
- Soporta STUN (Utilidades para sesión transversal de NAT) para usuarios detrás de un NAT (Traducción de direcciones de red).
- Soporte limitado para llamadas de video.

Otras funcionalidades de SipDroid se integran con la aplicación de marcación por defecto de Android por lo que el sistema consulta al usuario si opcionalmente desea hacer una llamada saliente usando SipDroid o a través de la red móvil contratada.

Desde su nacimiento, SipDroid se ha utilizado en muchas implementaciones SIP diferentes para aplicaciones en Android. De hecho, utiliza librerías para el manejo de protocolos específicos para transporte de flujos multimedia y es precisamente esta característica la que se ha aprovechado en este proyecto fin de carrera para el desarrollo de la aplicación.

A continuación se explica brevemente la configuración básica de la aplicación.

Una vez configurado el proveedor VoIP que se desee utilizar, se puede descargar la aplicación desde Google Play o directamente desde la siguiente URL:

http://code.google.com/p/sipdroid/

Una vez instalada, se abre la aplicación y se elige *Ajustes* como muestra la siguiente figura:

Figura 13. Pantalla principal de SipDroid.

Como se puede observar en la figura 14, hay posibilidad de configurar dos cuentas SIP. Se selecciona la primera y se añade el *Usuario de autorización* y la *Contraseña*, los cuales ya estaban configurados previamente en el cliente VoIP.

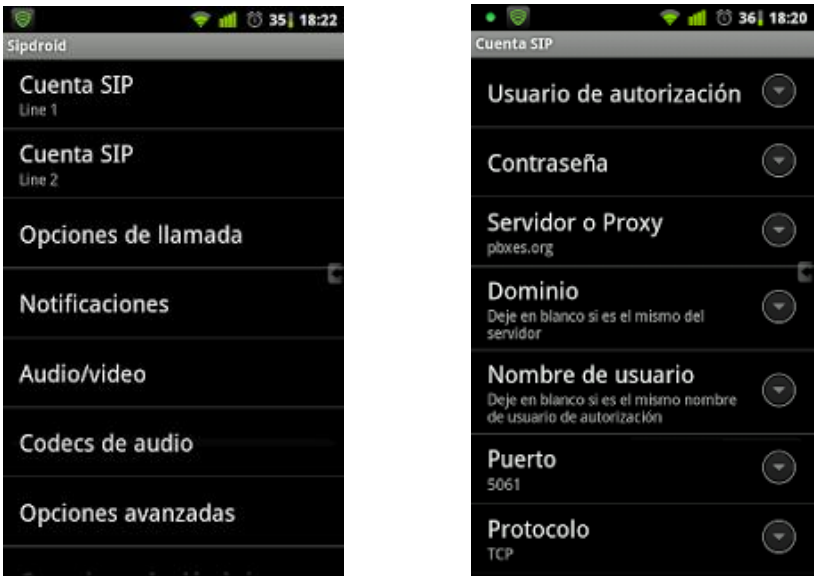

Figura 14. Pantallas de configuración de SipDroid.

También se puede elegir el tipo de red o el tipo de códec de voz para realizar las llamadas.

Después de introducir los datos anteriores se vuelve a la pantalla inicial. En el *cuadro Dirección de contraparte* se puede especificar la dirección de la cuenta VoIP a la que se pretende llamar.

3.2 APEINTA

3.2.1 Líneas generales del proyecto

APEINTA [63] es un proyecto español cuyo acrónimo significa *Apuesta Por la Enseñanza Inclusiva: uso de Nuevas Tecnologías dentro y fuera del Aula*. El objetivo principal es intentar apostar por una enseñanza educativa igualitaria que pueda integrar con mayor facilidad y eficiencia a la población con discapacidad utilizando como instrumentos las nuevas tecnologías tanto informáticas como telemáticas.

Dicha apuesta surge de la Universidad Carlos III de Madrid en colaboración con el Centro Español de Subtitulado y Audiodescripción y está financiado por el Gobierno de España.

La propuesta se basa en aportar facilidades dentro del aula a personas con discapacidad auditiva o locutiva, es decir, realizar una traducción de voz a texto o una traducción de texto a voz respectivamente. Además el proyecto dispone de una plataforma web donde se pueden encontrar multitud de contenidos para dar apoyo tanto a personas con discapacidad como a personas sin discapacidad, de este modo la información queda mucho más estructurada y accesible para todos los tipos de alumnos que lo precisen.

Por ello, se puede decir que el proyecto se ha desarrollado con dos focos de trabajo claramente diferenciados para facilitar el acceso al contenido de las asignaturas:

- **En el Aula.** Por un lado, tenemos una propuesta online dentro de la propia aula, de forma que mientras el profesor explica los contenidos de la charla existe un sistema de transcripción automática de voz a texto en tiempo real o en diferido para alumnos con discapacidad auditiva y además, un sistema de traducción texto a voz mediante mecanismos de síntesis de voz para personas con discapacidad locutiva o con problemas para expresarse en castellano.
- **Fuera del Aula.** Por otro lado, aporta otra solución promulgando una plataforma web donde colgar contenidos educativos en formato digital que pueden ayudar a todo tipo de personas en el aprendizaje de la materia en cuestión.

3.2.2 Arquitectura

La arquitectura del sistema se identifica en la figura 15 como una plataforma cliente-servidor en la que el servidor se encuentra en el ordenador del profesor y realiza básicamente dos tareas: la transcripción de la señal de voz y el envío del texto a los clientes representados por los dispositivos que tienen los estudiantes.

En primer lugar, un micrófono es usado por el profesor para registrar la voz de su discurso. A continuación se digitaliza la señal y se realiza el proceso computacional donde los algoritmos de un software ASR implementado en el servidor traducen los bits digitales a texto, generándose en tiempo real recursos educativos como subtítulos sincronizados o notas en diferentes formatos. El último paso es enviar esta información en forma de texto a los dispositivos finales de visualización de los alumnos, tales como pantallas de televisión, o puede

ser enviada a través de un servidor web a un ordenador portátil, agendas electrónicas, smartphones o gafas de subtitulado.

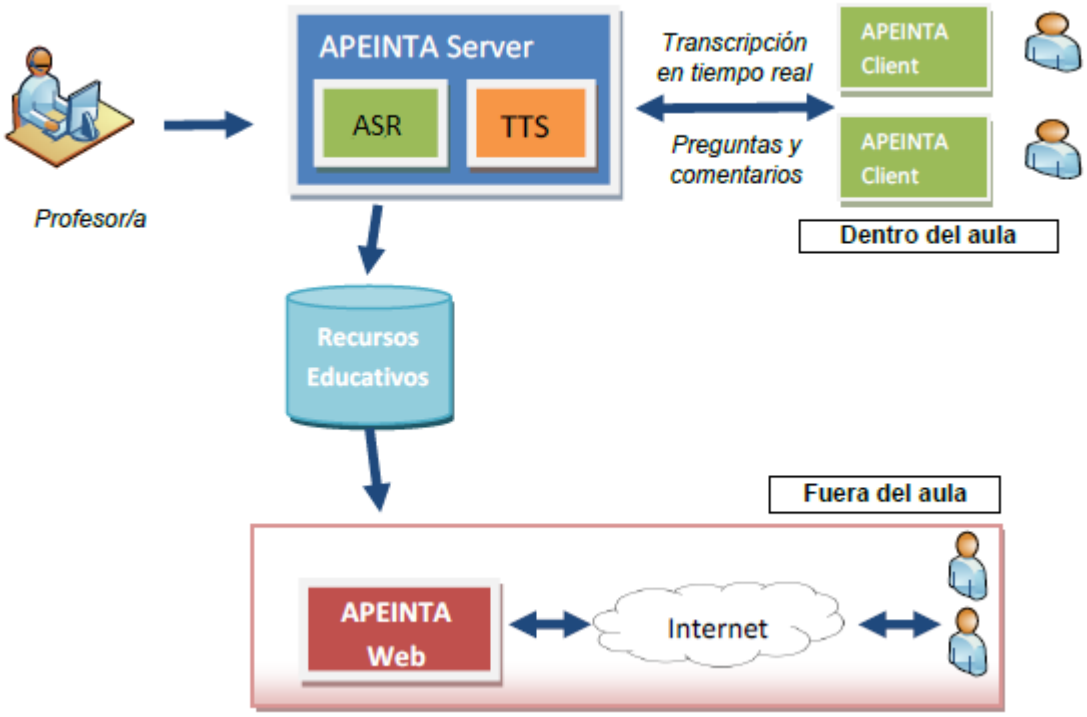

Figura 15 . Arquitectura original de APEINTA.

El ASR utilizado en la evaluación es el software comercial DNS v.10.1, capaz de transcribir habla continua y de gran vocabulario. Para implementar el motor ASR en el servidor se utilizó un SDK de DNS, programando un software en Visual C++ que comunica cliente y servidor mediante conexiones TCP (Protocolo de Control de Transmisión).

Paralelamente, el módulo de generación de voz sintética posibilita un canal de comunicación bidireccional, en el cual estudiantes con problemas de pronunciación o de idioma puedan plantear comentarios o dudas. Entonces, el servidor transforma en voz sintética estos mensajes de texto gracias al módulo TTS una vez que son recibidos desde los dispositivos de los estudiantes. Para implementar esta funcionalidad se ha usado *Speech API* (SAPI) de Microsoft, el cual puede modificar el tono, la velocidad y el volumen de voz que se utilizará.

Por último, la plataforma web de APEINTA proporciona acceso universal a los contenidos pedagógicos, vídeos de las conferencias, transparencias con la presentación de los profesores y otros documentos académicos. Por otra parte, los recursos educativos generados automáticamente durante las clases por el servicio de subtitulado en tiempo real como audio, subtítulos sincronizados y notas en diferentes formato, pueden ser publicados con facilidad en la plataforma como un recurso accesible y disponible para todos los estudiantes.

Además han sido consideradas normas como [AENOR 2003] y [AENOR 2004], iniciativas de accesibilidad web presentadas por el W3C (World Wide Web Consortium) [64] y estándares de accesibilidad como [IMS 2002] para el diseño en general de la plataforma web.

Capítulo 4. Análisis del Proyecto

4.1 Descripción global del sistema

El proyecto consiste en la realización de una aplicación de VoIP para dispositivos móviles con sistema operativo Android que debe cumplir una serie de requisitos definidos en una primera etapa de planteamiento del proyecto. Muchos de estos requisitos parten de una iniciativa de la Universidad Carlos III en colaboración con France Telecom España dentro del contexto de APEINTA para realizar un estudio de viabilidad para el envío de la voz del profesor por streaming hasta el servidor de subtitulado de APEINTA.

Con esta aplicación, que se ha decidido nombrar como *StreamDroid*, los profesores pueden enviar su propia voz en tiempo real como un flujo continuo de datos a través de Internet hasta el servidor de subtitulado. En el servidor de subtitulado se reproduce el flujo de audio y se procede a realizar la transcripción a texto, pero todas las actividades en dicho servidor no son objeto de estudio para definir los requisitos de la aplicación móvil.

A continuación, se pasan a detallar los distintos requisitos que debe cumplir la aplicación.

4.2 Requisitos de la aplicación StreamDroid

En esta sección se describen los requisitos de usuario y software que se han tenido en cuenta en el posterior diseño e implementación de StreamDroid partiendo de una primera fase de toma de requisitos. El objetivo es clasificarlos y definirlos formalmente en las fases de análisis y diseño de la aplicación, así como implementarlos en un prototipo y probar que se cumplen todas las funcionalidades requeridas.

Los requisitos de usuario y software proporcionan una visión completa del comportamiento del sistema y a su vez son especificaciones de las funcionalidades del sistema que han de ser resueltas.

4.2.1 Requisitos de Partida

Los campos para definir los requisitos de partida para el presente proyecto siguen la siguiente plantilla en forma de tabla:

Código:		Prioridad:	
Título:			
Descripción:			

Tabla 4. Plantilla para un Requisito de Partida.

Dónde:

- **Código**: identifica de forma única a cada requisito
  - RF- Requisito funcional
  - RNF- Requisito no funcional
  - Dentro de este segundo grupo, existen dos tipos:
    - RNFR- Requisito no funcional de rendimiento
    - RNFU- Requisito no funcional de usabilidad
- **Prioridad**: Importancia del requisito con respecto al resto:
  - Alta: Requisito básico para el funcionamiento de la aplicación
  - Media: Requisito opcional de prioridad 1
  - Baja: Requisito opcional de prioridad 2
- **Título**: Resumen de la definición del requisito
- **Descripción**: Detalle del requisito.

4.2.2 Análisis de Requisitos de Partida

Tal y como se ha descrito anteriormente, los requisitos de partida se han analizado, clasificado y definido formalmente en requisitos funcionales y no funcionales. A continuación se ofrece una descripción de cada uno de ellos a partir de la estructura definida en la Tabla 3.

4.2.2.1 Requisitos funcionales

Debido a la complejidad y número de requisitos funcionales, se ha decidido clasificarlos a partir de sus funcionalidades en 4 tipos:

- RF_AC: Requisitos funcionales de acceso a la aplicación
- RF_CF: Requisitos funcionales de configuración de la aplicación.
- RF_CG: Requisitos funcionales de control de la grabación.
- RF_PI: Requisitos funcionales de presentación de información por pantalla.

De acceso a la aplicación

Código:	RF_AC_01	Prioridad:	Alta
Título:	Instalación y acceso a la aplicación.		
Descripción:	La aplicación será accesible desde la lista general de aplicaciones del smartphone una vez instalada. No será necesario un registro inicial de usuario para su primer uso ni establecer ningún tipo de sesión de usuario.		

Tabla 5. Requisito RF_AC_01.

Código:	RF_AC_02	Prioridad:	Media
Título:	Salir de la aplicación.		
Descripción:	Permitir al usuario salir de la aplicación en el momento que quiera mediante algún elemento de la interfaz gráfica.		

Tabla 6. Requisito RF_AC_02.

De configuración de la aplicación

Código:	RF_CF_01	Prioridad:	Alta
Título:	Activar códec de voz sin compresión.		
Descripción:	Permitir la elección de un códec para voz sin compresión de datos y la posibilidad de poder habilitarlo para su utilización.		

Tabla 7. Requisito RF_CF_01.

Código:	RF_CF_02	Prioridad:	Media
Título:	Configurar códec de voz sin compresión.		
Descripción:	Permitir la configuración de un códec para voz sin compresión de datos. En función del tipo de códec, se podrán modificar opciones como: • Número de canales • Frecuencia de muestreo • Número de bits por muestra • Tasa de bits		

Tabla 8. Requisito RF_CF_02.

Código:	RF_CF_03	Prioridad:	Alta
Título:	Activar códec de voz con compresión.		
Descripción:	Permitir la elección de un códec para voz con compresión de datos y la posibilidad de poder habilitarlo para su utilización.		

Tabla 9. Requisito RF_CF_03.

Código:	RF_CF_04	Prioridad:	Media
Título:	Configurar códec de voz con compresión.		
Descripción:	Permitir la configuración de un códec de voz con compresión de datos. En función del tipo de códec, se podrán modificar opciones como: • Número de canales • Frecuencia de muestreo • Número de bits por muestra • Tasa de bits		

Tabla 10. Requisito RF_CF_04.

Código:	RF_CF_05	Prioridad:	Alta
Título:	Activar envío por streaming.		
Descripción:	Posibilitar la opción de enviar en tiempo real el flujo de voz que se graba desde el micrófono del smartphone a través de la conexión a Internet disponible.		

Tabla 11. Requisito RF_CF_05.

Código:	RF_CF_06	Prioridad:	Media
Título:	Selección de protocolo de transmisión.		
Descripción:	El usuario podrá seleccionar uno de los protocolos de transmisión de flujos multimedia disponibles en la aplicación, sin posibilidad de configurar más opciones y siempre que sea compatible con el códec de voz elegido.		

Tabla 12. Requisito RF_CF_06.

Código:	RF_CF_07	Prioridad:	Alta
Título:	Activar almacenamiento interno.		
Descripción:	Activar la opción de almacenar en local, es decir, en la memoria interna del smartphone, el flujo de voz que se graba desde el micrófono del smartphone para su posterior tratamiento.		

Tabla 13. Requisito RF_CF_07.

Código:	RF_CF_08	Prioridad:	Media
Título:	Configurar fichero para la grabación.		
Descripción:	Permitir elegir algunas opciones para el fichero que contiene la voz grabada como el nombre, la extensión o el directorio del sistema de almacenamiento interno donde se almacenará.		

Tabla 14. Requisito RF_CF_08.

Código:	RF_CF_09	Prioridad:	Alta
Título:	Configurar la conexión con servidor.		
Descripción:	Permitir la configuración de conexión remota con el servidor de subtitulado especificando opciones como: • Dirección IP • Puerto		

Tabla 15. Requisito RF_CF_09.

De control de la grabación

Código:	RF_CG_01	Prioridad:	Alta
Título:	Permitir iniciar grabación.		
Descripción:	El usuario podrá iniciar la grabación de su voz una vez que el códec de voz, protocolo de transmisión/almacenamiento y la conexión con el servidor se encuentren especificados. Se podrá llevar a cabo mediante la interacción con algún elemento de la interfaz gráfica.		

Tabla 16. Requisito RF_CG_01.

Código:	RF_CG_02	Prioridad:	Media
Título:	Permitir pausar grabación.		
Descripción:	El usuario podrá pausar la grabación de su voz siempre y cuando se encuentre ya activa en ese momento. Es necesaria la interacción con un elemento de la interfaz para ejecutar la acción.		

Tabla 17. Requisito RF_CG_02.

Código:	RF_CG_03	Prioridad:	Media
Título:	Permitir continuar grabación.		
Descripción:	El usuario podrá proseguir con la grabación de su voz si se encontraba pausada en el momento de tomar la decisión. La acción se realizará mediante la interacción con algún elemento de la interfaz gráfica.		

Tabla 18. Requisito RF_CG_03.

Código:	RF_CG_04	Prioridad:	Alta
Título:	Permitir finalizar grabación.		
Descripción:	El usuario podrá finalizar la grabación de su voz cuando lo desee siempre que se encuentre en ese instante grabando. Algún elemento de la interfaz gráfica posibilitará dicha acción.		

Tabla 19. Requisito RF_CG_04.

De presentación de información por pantalla.

Código:	RF_PI_01	Prioridad:	Media
Título:	Mostrar información del códec.		
Descripción:	Se especificará en algún lugar de la pantalla de grabación de la voz, información específica acerca del códec usado y la configuración elegida para el mismo.		

Tabla 20. Requisito RF_PI_01.

Código:	RF_PI_02	Prioridad:	Media
Título:	Mostrar información del transporte/almacenamiento interno.		
Descripción:	Se indicará en la pantalla de grabación de la voz, información específica acerca del protocolo de transmisión usado para el streaming o si se la grabación se almacena localmente, según lo configurado por el usuario.		

Tabla 21. Requisito RF_PI_02.

Código:	RF_PI_03	Prioridad:	Alta
Título:	Mostrar evolución temporal de la grabación.		
Descripción:	Será necesario indicar el tiempo transcurrido de grabación en la pantalla habilitada para su control teniendo en cuenta las posibles pausas y otras acciones que realice el usuario.		

Tabla 22. Requisito RF_PI_03.

4.2.2.2 Requisitos no funcionales de rendimiento

Código:	RNFR_01	Prioridad:	Alta
Título:	Conexión a Internet.		
Descripción:	Permitir que la aplicación pueda manejar datos de redes móviles 2G, 3G o 4G o de redes WiFi cuando se requiera el envío de la voz por streaming.		

Tabla 23. Requisito RNFR_01.

Código:	RNFR_02	Prioridad:	Alta
Título:	Disponibilidad total.		
Descripción:	El servicio deberá estar disponible siempre que el usuario lo necesite. La aplicación deberá seguir ejecutando los servicios correspondientes mientras el usuario no elimine el proceso manualmente.		

Tabla 24. Requisito RNFR_02.

Código:	RNFR_03	Prioridad:	Media
Título:	Persistencia de los datos de configuración.		
Descripción:	Permitir que los datos de configuración del usuario (tipo de códec, protocolos de transmisión, conexión con servidor…) se encuentren disponibles de forma permanente.		

Tabla 25. Requisito RNFR_03.

Código:	RNFR_04	Prioridad:	Alta
Título:	Resolución de la aplicación.		
Descripción:	La aplicación se adaptará a la resolución del dispositivo en que se ejecute y al tamaño de pantalla.		

Tabla 26. Requisito RNFR_04.

Código:	RNFR_05	Prioridad:	Baja
Título:	Tiempo de respuesta general.		
Descripción:	El tiempo de respuesta general de la aplicación al mostrar cualquier pantalla deberá ser menor o igual a dos segundos.		

Tabla 27. Requisito RNFR_05.

4.2.2.3 Requisitos no funcionales de usabilidad

Código:	RNFU_01	Prioridad:	Alta
Título:	Licencia GPL.		
Descripción:	El código de la aplicación desarrollada será de libre distribución, gozando de licencia GNU GPL.		

Tabla 28. Requisito RNFU_01.

Código:	RNFU_02	Prioridad:	Alta
Título:	Versión de Android.		
Descripción:	La aplicación será ejecutable sólo en sistemas móviles Android, versión 2.2 o superior.		

Tabla 29. Requisito RNFU_02.

Código:	RNFU_03	Prioridad:	Media
Título:	Lenguaje sencillo.		
Descripción:	Se usará un lenguaje sencillo para que sea cercano a los usuarios.		

Tabla 30. Requisito RNFU_03.

Código:	RNFU_04	Prioridad:	Baja
Título:	Evitar que los usuarios introduzcan mucho texto.		
Descripción:	Se deberán proporcionar mecanismos al usuario para que deba introducir la menor cantidad de texto posible donde corresponda.		

Tabla 31. Requisito RNFU_04.

Código:	RNFU_05	Prioridad:	Baja
Título:	Información importante en la parte superior.		
Descripción:	La información importante deberá ser simple y tiene que ser mostrada en la parte superior de la pantalla.		

Tabla 32. Requisito RNFU_05.

Código:	RNFU_06	Prioridad:	Media
Título:	Permitir alertas.		
Descripción:	Se mostrarán alertas mediante ventanas emergentes sobre errores, sugerencias, nuevo estado de la grabación etc.		

Tabla 33. Requisito RNFU_06.

Capítulo 5. Diseño del Proyecto

En la sección anterior se han listado y descrito los requisitos de usuario y software de partida para StreamDroid, la aplicación para Android que surgió de la necesidad de enviar voz por streaming hasta el servidor de subtitulado de APEINTA. El objetivo del análisis del sistema no ha sido otro que la obtención de una especificación detallada de las características y funcionalidades de la aplicación para así tener una base sólida sobre la que construir el diseño y posterior implementación. Por ello, esta nueva sección trata de interpretar los requisitos identificados en la fase anterior para llevar a cabo un diseño correcto de la aplicación móvil a implementar.

Se debe prestar especial atención a los requisitos de diseño del sistema para que su incorporación al mismo se realice de manera satisfactoria. Una mala elección de requisitos funcionales en la etapa de diseño puede conducir a un fracaso en términos de aceptación de la aplicación por los usuarios y objetivos finales. Además, durante el desarrollo de este apartado se detallará la correspondencia entre los requisitos de análisis y los requisitos de diseño.

5.1 Diseño funcional de StreamDroid

Este apartado tiene como fin detallar todas las funcionalidades específicas y genéricas de la aplicación, dando sentido por tanto a la necesidad de la misma para cubrir los objetivos de este proyecto. Por ello, se han especificado una serie de requisitos funcionales de aplicación (RFA) que han sido abordados para las fases de diseño e implementación.

Cada uno de estos requisitos se define con una tabla que sigue la siguiente plantilla:

Código:	
Título:	
Requisito relacionado:	

Tabla 34. Plantilla para un Requisito Funcional de aplicación.

Dónde:

- **Código**: identifica de forma única cada requisito
- **Título**: resumen de la definición del requisito
- **Requisito relacionado**: requisito/s de usuario o software identificados en la fase de análisis y que guardan relación con el RFA a analizar. Es importante destacar que un único RFA puede estar relacionado con varios requisitos de análisis, ya sean funcionales o no funcionales de usabilidad o rendimiento, y que resultan necesarios todos ellos para poder realizar correctamente el diseño funcional del RFA concreto.

De forma adicional a la especificación de cada RFA con la tabla anterior, se describe de forma clara y detallada el objetivo del mismo.

Código:	RFA_01
Título:	Acceso a la aplicación.
Requisito relacionado:	RF_AC_01, RNFU_01,RNFU_02

Tabla 35. Requisito RFA_01.

Detalle del requisito:

Una vez instalada la aplicación en el smartphone, para lo cual será necesaria una versión de Android igual o superior a 2.2, se puede acceder a la misma a través del menú o lista habitual de aplicaciones seleccionando el icono correspondiente a *StreamDroid*.

Se ha prescindido de un tipo de acceso por usuario con registro previo puesto que la utilización de la aplicación no requiere de un perfil concreto de usuario.

Código:	RFA_02
Título:	Pantalla principal de la aplicación.
Requisito relacionado:	RF_PI_01, RF_PI_02, RF_PI_03, RNFR_04, RNFR_05, RNFU_05, RNFU_06

Tabla 36. Requisito RFA_02.

Detalle del requisito:

La pantalla principal de aplicación aparecerá tras iniciar la misma y se dividirá en tres áreas bien diferenciadas de arriba a abajo:

- Información multimedia: texto informativo con el códec usado y el tratamiento de la voz grabada mediante el envío por streaming o almacenamiento en local.
- Cronómetro que indica el tiempo de grabación en cualquier instante.
- Botones táctiles para controlar la grabación.

La interfaz se adaptará a la resolución y tamaño de la pantalla independientemente del dispositivo desde el cual se maneje la aplicación.

Además, se podrá acceder al menú de aplicación donde aparecerán dos opciones:

- Acceso a la pantalla de configuración.
- Opción para salir de la aplicación.

Código:	RFA_03
Título:	Pantalla general de configuración.
Requisito relacionado:	RNFR_08, RNFR_09, RNFU_03, RNFU_04, RNFU_06

Tabla 37. Requisito RFA_03.

Detalle del requisito:

Se accederá desde la opción correspondiente en el menú de aplicación y su principal función será permitir el acceso a todas las configuraciones de la aplicación como:

- Configuración de streaming de audio sin compresión.
- Configuración de streaming de audio con compresión.
- Configuración de almacenamiento en local de audio sin compresión.
- Configuración de la conexión con el servidor.

A excepción de los ajustes para la conexión con el servidor, el resto de configuraciones dispondrán de su propia pantalla individual para manejar todas las opciones.

Se ha decidido por no añadir complejidad a la aplicación, que los dos tipos de audio que maneja la aplicación, con compresión y sin compresión, sean transportados por streaming mediante un protocolo de red específico distinto para cada uno, de forma que una vez que se haya seleccionado el tipo de códec, el protocolo para la transmisión también quedará fijado.

Código:	RFA_04
Título:	Envío streaming de audio sin compresión.
Requisito relacionado:	RF_CF_01, RF_CF_02, RF_CF_05, RF_CF_06, RNFR_03, RNFU_04

Tabla 38. Requisito RFA_04.

Detalle del requisito:

Esta pantalla de ajustes será accesible desde la pantalla general de configuración, y su principal funcionalidad será la activación del envío de voz sin compresión por streaming. Además, el usuario podrá manejar dos opciones para el códec:

- Número de canales: mono/estéreo.
- Frecuencia de muestreo en Hz.

Código:	RFA_05
Título:	Envío streaming de audio con compresión.
Requisito relacionado:	RF_CF_03, RF_CF_04, RF_CF_05, RF_CF_06, RNFR_03, RNFU_04

Tabla 39. Requisito RFA_05.

Detalle del requisito:

De la misma forma que RFA_04, la pantalla para activar el envío de streaming de audio con compresión, únicamente será accesible desde la pantalla general de configuración. También dispondrá de una serie de opciones para el códec y protocolo de transporte:

- Tasa de codificación en Kbps para el códec.
- Tamaño de *payload* (carga útil de un paquete de datos) en bytes.
- Posibilidad de usar las librerías nativas o de terceros para el soporte al protocolo de transporte elegido. Esta doble posibilidad se debe al hecho de que para versiones de Android anteriores a 3.1, no existe soporte nativo y era necesario el uso de librerías externas. Además, la opción de cambiar el tamaño de payload solo estará disponible usando las librerías de terceros.

Código:	RFA_06
Título:	Almacenamiento local de audio con compresión.
Requisito relacionado:	RF_CF_07, RF_CF_08, RNFR_03, RNFU_04, RNFU_06

Tabla 40. Requisito RFA_06.

Detalle del requisito:

Al igual que RFA_04 y RFA_05, la única forma de acceder a esta nueva pantalla de ajustes será desde la pantalla general de configuración. Además de activar la grabación en local de un fichero de audio con la voz, el usuario tendrá la posibilidad de elegir:

- Tasa de codificación en Kbps para el códec.
- Nombre del directorio para el fichero.
- Nombre del fichero de audio, incluyendo extensión.

Es importante destacar que si una de las dos últimas características no quedará definidas, el usuario sería avisado con una alerta por pantalla.

Código:	RFA_07
Título:	Conexión con servidor.
Requisito relacionado:	RF_CF_09, RNFR_03, RNFU_04, RNFU_06

Tabla 41. Requisito RFA_07.

Detalle del requisito:

Para la conexión con el servidor, el usuario podrá modificar dos opciones directamente desde la pantalla general de configuración no siendo necesario el acceso a una pantalla de ajustes independiente:

- Nombre o dirección IP del servidor.
- Puerto de destino en el servidor.

En ambos casos será imprescindible dejar indicados los datos correctamente, pues de ello dependerá que el flujo de voz sea recibido correctamente en el servidor. Se han diseñado alertas por pantalla para el caso de que estos dos campos no sean especificados.

Código:	RFA_08
Título:	Servicio para gestionar la grabación.
Requisito relacionado:	RNFR_01, RNFR_02, RF_PI_01, RF_PI_02

Tabla 42. Requisito RFA_08.

Detalle del requisito:

Se deberá implementar un servicio que será el encargado de gestionar y controlar todas las operaciones en segundo plano durante el proceso de la grabación de la voz y envío por streaming a través de internet:

1. Crear el servicio especificando los parámetros del códec de voz y del protocolo de transmisión relacionado que haya configurado el usuario previamente en caso de que sea haya seleccionado el envío por streaming.
2. Comprobar que el smartphone tiene conectividad con internet para poder realizar la comunicación con el servidor.

3. Finalizar por completo la configuración del servicio (especificación de todas las características del tipo de códec usado para la voz, adecuación de los datos al protocolo de transmisión a nivel de paquete o flujo de datos, conexión con servidor etc) comprobado además que tanto el software como el hardware del dispositivo son adecuados para iniciar el mismo.
4. Lanzar el servicio si se cumplieron todos los requisitos de estados anteriores.
5. Mantener en segundo plano el servicio sin cortes ni errores independientemente del estado visible de la aplicación por parte del usuario y de las acciones que éste pudiera realizar.
6. Finalización de servicio y liberación de recursos para el sistema cuando el usuario decida terminar con el proceso de grabación y envío de audio.

A continuación se muestra el diagrama de estados por los que debe pasa el servicio definido con anterioridad:

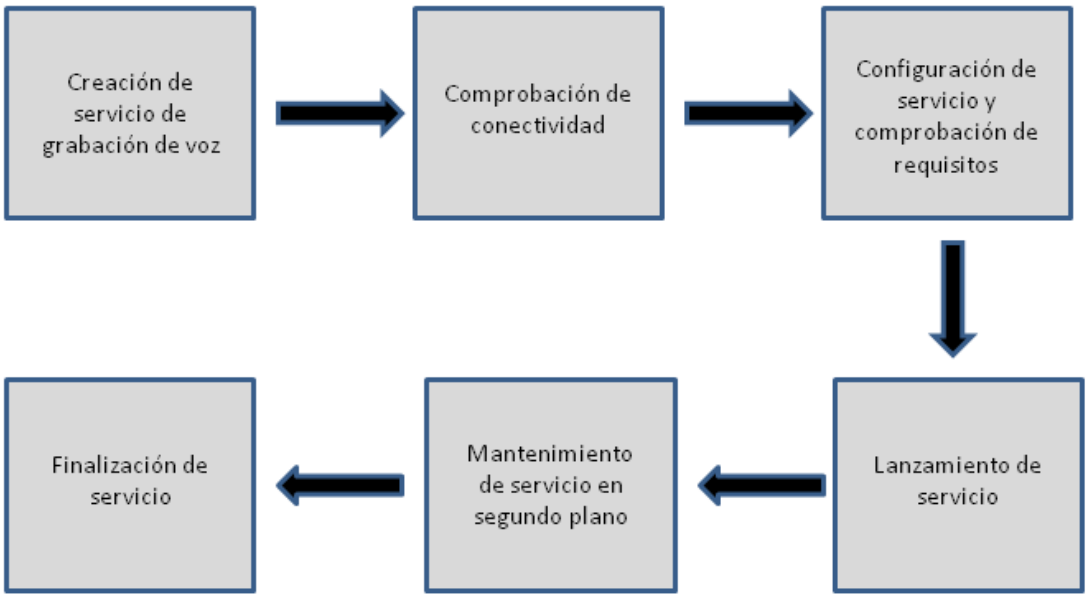

Figura 16. Diagrama de Estados para el servicio de gestión de grabación.

Código:	RFA_09
Título:	Control de grabación.
Requisito relacionado:	RF_CG_01, RF_CG_02, RF_CG_03, RF_CG_04, RF_PI_03

Tabla 43. Requisito RFA_09.

Detalle del requisito:

Se dispondrá de tres elementos gráficos en la pantalla principal de aplicación para realizar todas las operaciones sobre el control de la grabación y su estado temporal:

- Cronómetro: Indicar el tiempo transcurrido de grabación. Podría cambiar el color del texto en función del estado de la grabación (en pausa o en ejecución).
- Botón 1: Iniciar, pausar o continuar la grabación.
- Botón 2: Resetear la grabación, es decir, finalizar la grabación y pone a cero el contador de tiempo transcurrido en el cronómetro.

Código:	RFA_10
Título:	Control de errores por problemas de conectividad.
Requisito relacionado:	RNFR_01, RNFU_06

Tabla 44. Requisito RFA_10.

Detalle del requisito:

En caso de existir un problema de conectividad con internet en el momento anterior a iniciar el servicio de grabación de voz (ver diagrama de estados de RFA_08), la aplicación lanzará una alerta gráfica avisando al usuario de que revise la configuración o la conexión a internet del dispositivo.

Código:	RFA_11
Título:	Salir de la aplicación.
Requisito relacionado:	RF_AC_02

Tabla 45. Requisito RFA_11.

Detalle del requisito:

Se accederá desde la opción correspondiente en el menú de aplicación y su única función será indicar al sistema operativo que deberá finalizar el proceso asociado a la aplicación, de forma que todos los servicios en segundo plano también se finalizarán.

5.2 Diagrama de bloques de StreamDroid

Se ha diseñado un diagrama de los bloques principales de la aplicación móvil StreamDroid de cara a poder implementar posteriormente cada bloque funcional por separado según la arquitectura de Android y las posibilidades que ofrece su framework de desarrollo. En la siguiente figura se puede visualizar el diagrama:

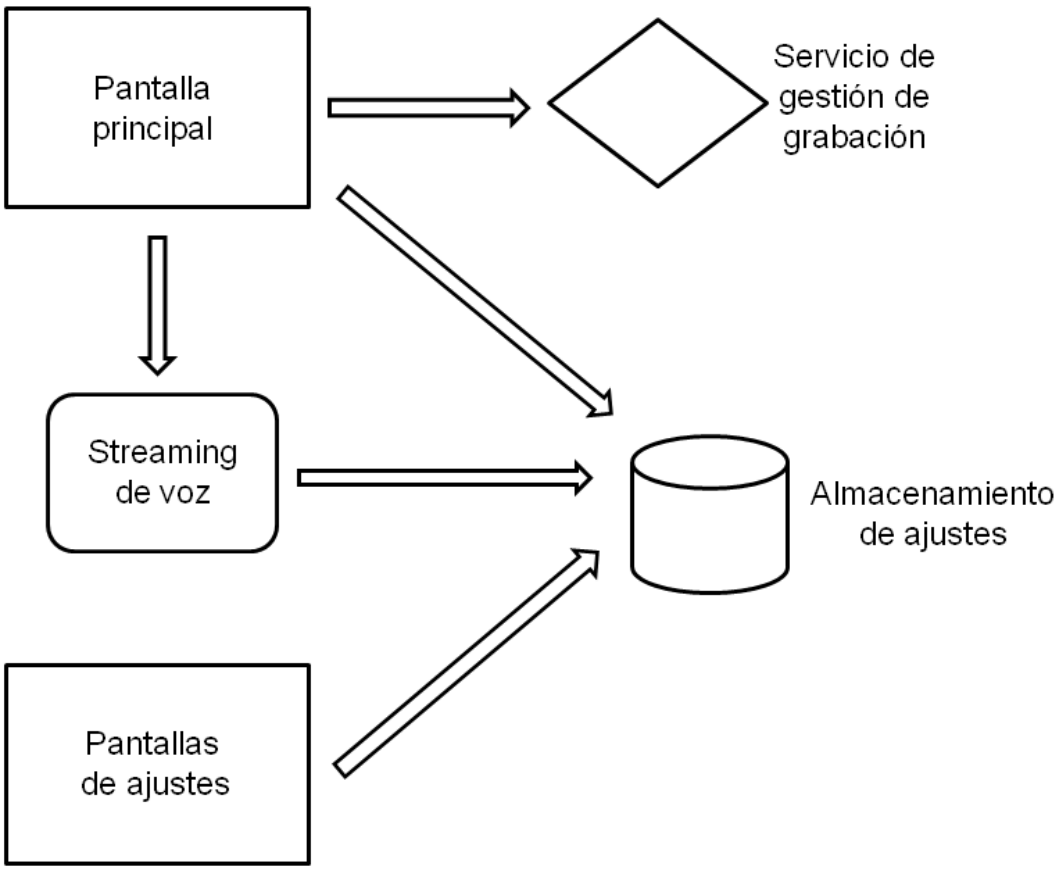

Figura 17. Diagrama de bloques funcionales de StreamDroid.

- **Pantalla principal:** Desde la cual se tiene acceso a los elementos de control del *Servicio de gestión de grabación*. Además se ha de especificar el tipo de streaming de voz.
- **Servicio de gestión de la grabación:** Sólo depende de los botones de control disponibles en la *Pantalla principal*.
- **Streaming de voz:** Necesita algunas variables de configuración como el tipo de códec y su formato, así como el tipo de tratamiento que se le aplicará: envío por internet o almacenamiento en local.
- **Pantallas de ajustes:** se refiere a la pantalla principal de configuración, así como todas las secciones de configuración restantes. Todas estas interfaces necesitan consultar constantemente el *Almacenamiento de ajustes* para mostrar la información al usuario.
- **Almacenamiento de ajustes:** Se trata de una memoria interna con el estado de todas las variables de configuración de la aplicación. El resto de bloques necesitan consultar el estado de estas variables para poder funcionar.

5.3 Diseño de la Arquitectura del sistema

La arquitectura del sistema está basada en una comunicación cliente-servidor. Por tanto, el sistema presenta dos subsistemas claramente diferenciados: un cliente en forma de aplicación móvil, y un servidor de subtitulado en tiempo real.

La siguiente figura representa la arquitectura general del sistema de subtitulado en tiempo real de APEINTA, descrita ya en la sección 3.2 de este documento:

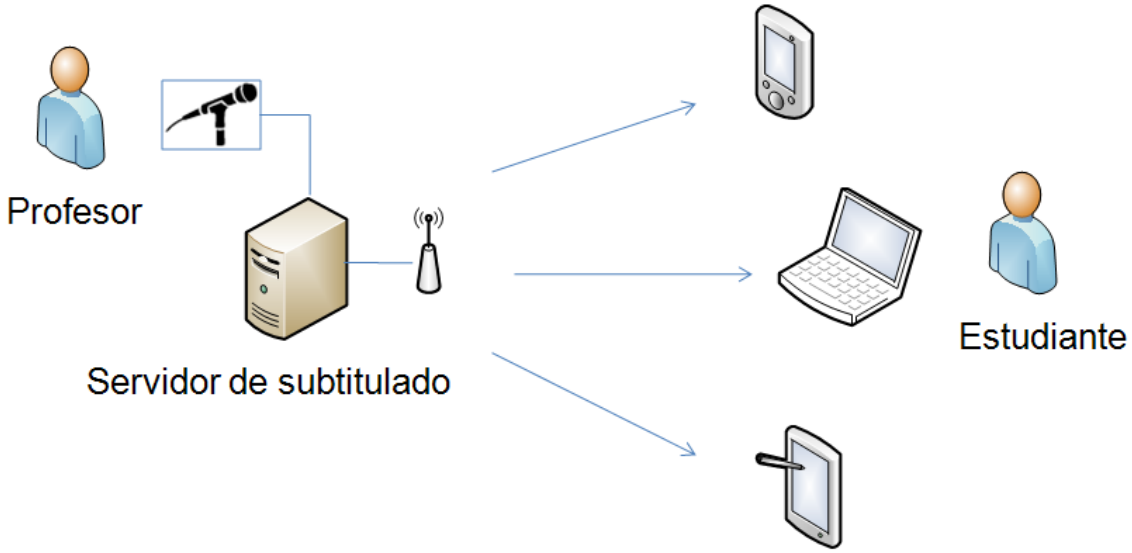

Figura 18. Arquitectura actual para el sistema de subtitulado APEINTA.

En base a los estudios y pruebas realizadas hasta ahora, la arquitectura final del servicio de transcripción en tiempo real propuesto para este proyecto trata de sustituir el micrófono convencional que manipula el profesor para grabar su propia voz por un terminal móvil con conexión de datos como se puede ver en la siguiente figura:

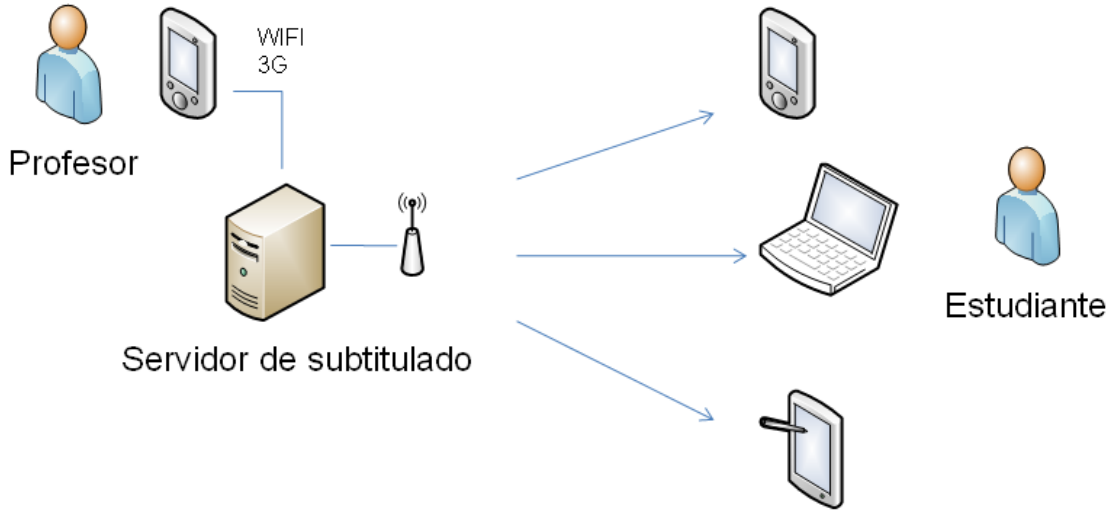

Figura 19. Arquitectura para el nuevo sistema de subtitulado propuesto.

Por tanto, se ha optado por una aplicación desarrollada para terminales Android, *StreamDroid*, la cual graba voz en un flujo de audio sin comprimir o usando algún códec con compresión. Después, este flujo de audio se transmite por streaming sobre un protocolo de transporte multimedia hasta el servidor de subtitulado en tiempo real de APEINTA.

Finalmente en dicho servidor el reproductor multimedia interpreta el streaming de audio y mediante el uso de algún mecanismo que permita extraer el audio reproducido y encaminarlo directamente al software de ASR usado, DNS, se consigue transcribir el audio digital a texto. Idealmente, el servidor de subtitulado dispone de dos tarjetas de sonido en paralelo para llevar a cabo el mecanismo descrito. En caso de que el servidor no disponga de este hardware, en uno de los anexos de este documento se proporciona una Guía de Usuario donde se propone alguna alternativa.

Además la comunicación entre el terminal móvil y el servidor de subtitulado, al margen de los protocolos empleados, se realiza a través de VoIP conectando el terminal móvil a una red de datos 3G o mediante una red WiFi disponible a la que haya posibilidad de conexión.

La siguiente figura (figura 20) ilustra el módulo de comunicación entre el terminal móvil y el servidor de transcripción dentro de la arquitectura general. Los nuevos componentes que se añaden con respecto a la arquitectura original de APEINTA (ver figura 18) son el terminal móvil con StreamDroid, un proceso de recepción de streaming y un sistema de conversión de datos.

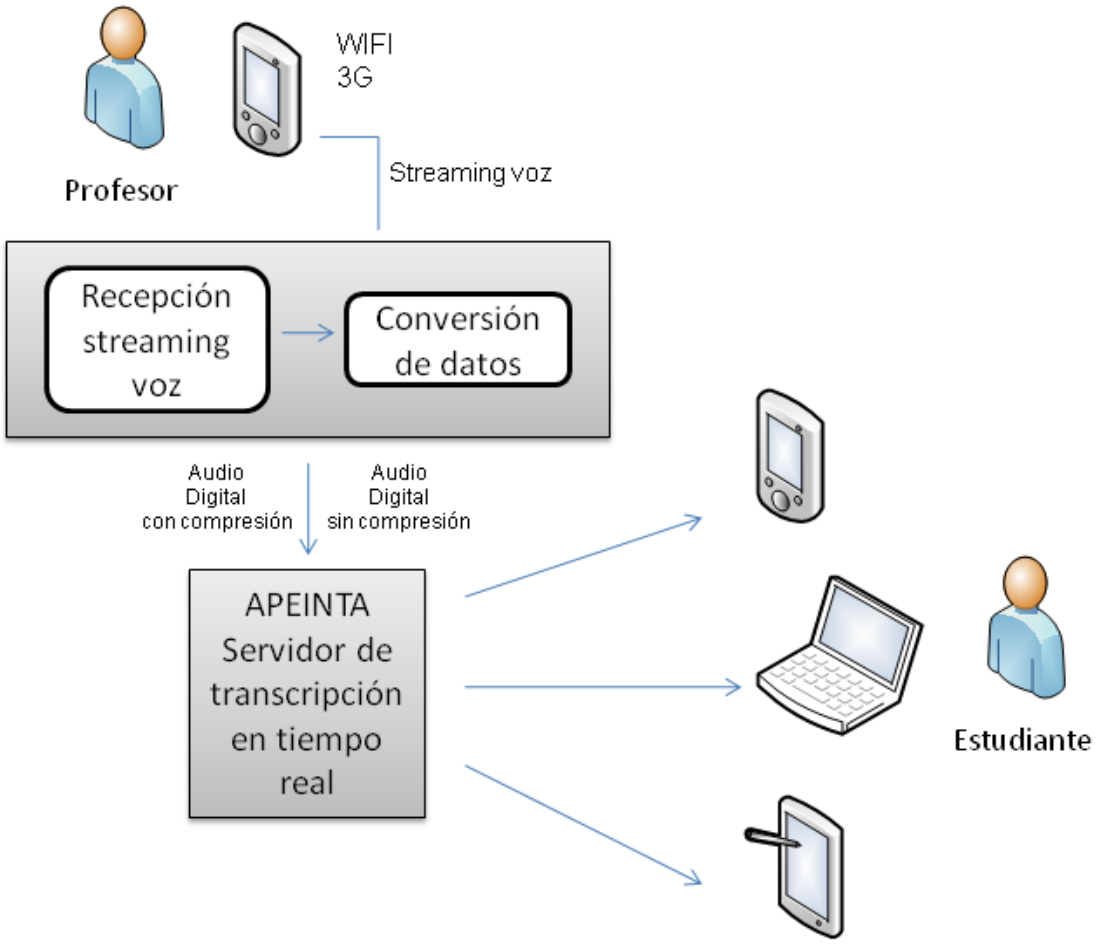

Figura 20. Módulo de comunicación entre el terminal móvil y el servidor de transcripción.

Capítulo 6. Implementación

Tras la especificación de todos los requisitos funcionales de la aplicación *SreamDroid*, en este apartado se describirá cómo se ha realizado la implementación final así como la tecnología empleada en el desarrollo de la aplicación móvil. También se detallarán algunos aspectos de la implementación en el servidor de subtitulado.

Además, se realizará previamente un estudio genérico de diferentes tecnologías para implementar algunas funcionalidades básicas del sistema como la codificación de la voz o el transporte y tratamiento del streaming de audio, comparando diferentes alternativas y justificando finalmente la que se ha elegido.

6.1 Codificación de la Voz

6.1.1 Introducción

La investigación en el campo de la codificación de la voz comenzó hace más de medio siglo, surgida por la necesidad de transmitir voz por los cables de telegrafía de pequeño ancho de banda. Antes de la Segunda Guerra Mundial un investigador de los laboratorios Bell, Homer Dudley, introdujo por primera vez la idea de codificador de voz o *vocoder* y durante la década de los 40 el muestreo de voz cobró fuerza gracias a las posibilidades de los sistemas digitales.

El surgimiento de la tecnología VLSI (integración a escala muy grande) durante los años 60 y 70, permitió nuevas soluciones como estudios basados en la Transformada de Fourier [65].

Durante las siguientes décadas, la investigación ha tenido como objetivo conseguir codificadores que utilicen un menor ancho de banda pero mejorando a la vez la calidad de la voz. Además gracias a esto se permite aprovechar con más eficiencia y eficacia los canales de transmisión y sistemas de almacenamiento y se facilita la encriptación.

A continuación, se realiza un estudio de los principales códecs y conceptos básicos de la codificación de la voz dada la importancia de elegir correctamente la naturaleza del audio para el sistema de subtitulado presentado en este proyecto.

6.1.2 Conceptos básicos

El objetivo fundamental de la codificación de voz es la conversión de la señal de voz a una secuencia binaria o representación digital, lo cual resulta imprescindible para poder enviar los datos por streaming o cualquier otra tecnología de transmisión.

La señal de voz es de carácter analógico, es decir, es una señal continua en tiempo y amplitud, por lo que su codificación conlleva un proceso básico de muestreo en tiempo y cuantificación en amplitud para conseguir una representación digital mediante la conversión A/D (Analógico-Digital). Para que en este proceso de digitalización no exista pérdida de información, se debe

muestrear la señal a una velocidad o frecuencia de muestreo (F_M) que como mínimo sea el doble de la frecuencia más alta presente en la señal según el Teorema de *Nyquist*.

En el proceso de cuantificación en amplitud se deben usar un número de bits por muestra (N) en función de la calidad que se pretenda obtener (resolución). Así, por ejemplo, una señal de voz con calidad telefónica tiene una frecuencia máxima de 4 KHz lo que supone una frecuencia de muestreo mínima de 8 KHz (8000 muestras por segundo) y suele utilizarse una representación de 8 bits por cada muestra (256 niveles de cuantificación), lo que conlleva a una velocidad de transmisión de 64 Kbps, conocida como DS0 (Señal digital 0).

Por último, el proceso de codificación, como tal, toma como señal de entrada la obtenida de la conversión A/D a una velocidad de (F_M x N) bits/segundo y utilizando ciertas propiedades de la señal de voz obtiene una nueva codificación con una velocidad de R bits/segundo que siempre será inferior a la inicial.

6.1.3 Audio digital: PCM

La modulación por impulsos codificados (MIC o PCM por sus siglas inglesas de *Pulse Code Modulation*) es un método de modulación patentado por Alec Reeves en 1937 para transformar una señal analógica en una señal digital o secuencia de bits.

Es el formato estándar de audio digital en los ordenadores así como en otros entornos como en los sistemas digitales de telefonía.

6.1.3.1 Proceso de muestreo y cuantificación

Los flujos de datos en PCM tienen dos propiedades básicas que determinan su fidelidad con respecto a la señal analógica original: la F_M, o número de veces por segundo que se toman las muestras, y la profundidad de bits o resolución, lo que determina el rango de posibles valores digitales codificados en binario que identifican a cada muestra.

En la siguiente figura se puede observar el muestreo y cuantificación de una onda senoidal:

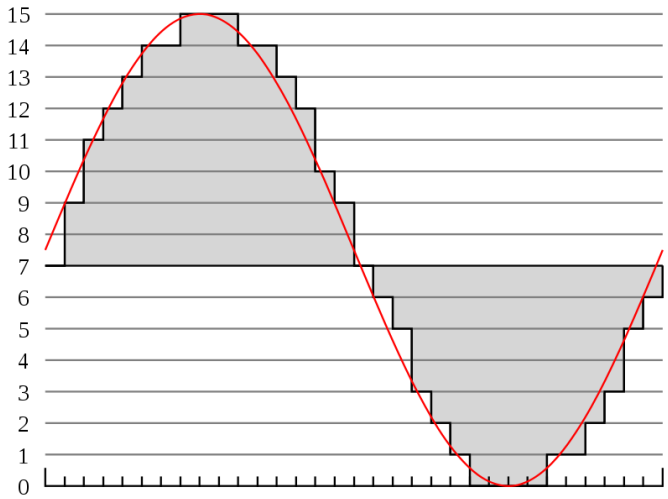

Figura 21. Muestreo y cuantificación de una onda senoidal en código PCM de 4 bits.

Los segmentos mostrados sobre el eje de abscisas representan los intervalos de tiempo que se usan para tomar las muestras mientras que las marcas sobre el eje de ordenadas indican los 16 valores posibles que puede tomar cada una debido a que la codificación usada es de 4 bits.

El resultado de este proceso es un código binario cuyo tamaño dependerá de la F_M y de la resolución y que además puede ser fácilmente manipulado para almacenarlo en un códec determinado para su distribución.

En la figura 22 se muestra un sistema que utiliza PCM para la codificación de señales y su transmisión por tres canales. Para la recuperación de la onda original (a), (b) y (c) se aplica un proceso inverso, obteniendo bastantes buenos resultados a pesar de la distorsión o ruido de cuantificación que se produce durante el proceso de cuantificación, debido al redondeo de las muestras a los valores cuánticos. Para minimizar al máximo esta distorsión, los sistemas normalizados eligen intervalos de cuantificación específicos.

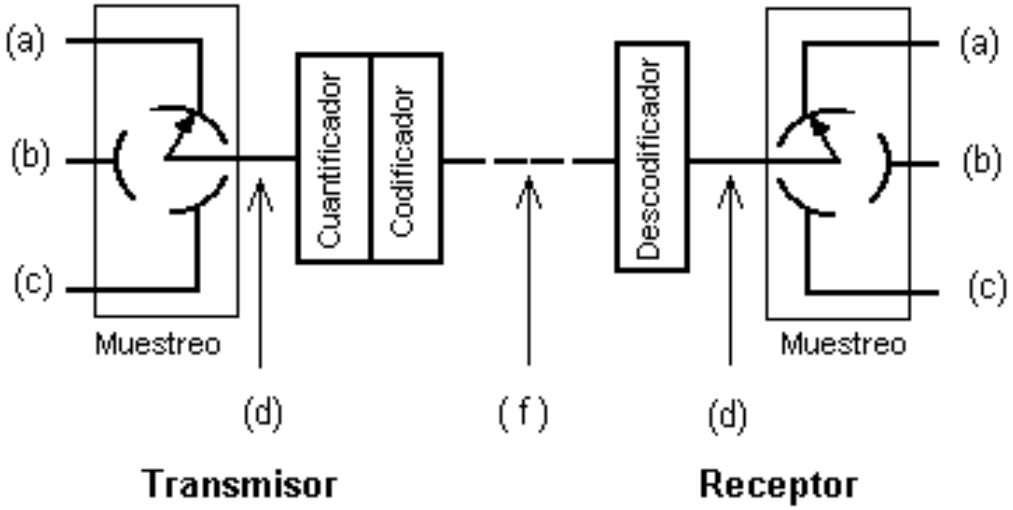

Figura 22. Disposición de elementos en un sistema PCM de tres canales.

6.1.3.2 Sistemas derivados para telefonía móvil

La CCITT (Comité Consultivo Internacional Telegráfico y Telefónico), actualmente conocida como UIT-T o Sector de Normalización de las Telecomunicaciones de la UIT (Unión Internacional de Telecomunicaciones) [66], propuso como estándares algunos nuevos sistemas surgidos a partir del PCM [67]:

- **DPCM** (Modulación por impulsos codificados diferencial) codifica los valores de la señal PCM como las diferencias entre el actual y el valor a predecir. En concreto, un algoritmo predice la siguiente muestra en base a las muestras anteriores, y el codificador almacena sólo la diferencia entre la predicción y el valor real. En el caso del audio, se reduce el número de bits por muestra al 25% en relación al PCM tradicional.
- **ADPCM** (DPCM adaptativo) es una variante de DPCM que permite una reducción del ancho de banda necesario variando el tamaño de los intervalos de cuantificación. De esta forma, consigue codificar canales telefónicos con velocidades de 16, 24 y 32 Kbps.
- **Modulación Delta** es otra variante de DPCM que utiliza un bit por muestra.

6.1.4 Códecs

La voz debe ser codificada para poder transmitirse a través de la red IP. Para ello se hace uso de códecs que garanticen la codificación y compresión del audio.

Un códec de voz es un conjunto de algoritmos [68] que permiten codificar y decodificar los datos auditivos de la voz humana, lo cual implica reducir la cantidad de bits iniciales sin modificar sustancialmente la información reconocible. Sirve para comprimir señales o ficheros de audio y posteriormente descomprimir los datos resultantes para reproducirlos o manipularlos en un formato más apropiado consiguiendo una buena calidad final. Se implementa en software, hardware o una combinación de ambos.

6.1.4.1 Parámetros

Existen una serie de parámetros que caracterizan a los códecs de audio:

- **Número de canales**: hace referencia al número de señales de audio simultáneas que pueden ser transportados en un flujo de audio. En función del número de canales, puede ser mono (1 canal), estéreo (2 canales) o multicanal.
- **F_M**: determina la calidad percibida, por lo tanto cuanto más alto sea mayor será la fidelidad del sonido obtenido respecto al original. Puesto que el sistema auditivo humano no es capaz de percibir frecuencias superiores a 20 KHz y cumpliendo a su vez el criterio de Nyquist, nunca se deben usar F_M superiores a 44.1 KHz.
- **Número de bits por muestra**: determina la precisión con la que se reproduce la señal original y el rango dinámico de la misma. Se suelen utilizar 8 (para un rango dinámico de hasta 45 dB), 16 (para un rango dinámico de hasta 90 dB como el formato CD) o 24 bits por muestra (para 109 a 120 dB de rango dinámico). El más común es 16 bits.
- **Tasa de bits**: representa el número de bits de información por unidad de tiempo. Puede ser constante (CBR), variable (VBR) o en media (ABR). En audio suele optarse por una VBR, especialmente cuando hay silencios o segmentos con pocas muestras.
- **MOS** (Factor de calidad): es la opinión conceptual de calificación que proporciona una medida numérica de calidad de la voz humana en el destino. El esquema utiliza pruebas subjetivas (medidas de opinión) que son calculadas matemáticamente, obteniendo un indicador cuantitativo de las cualidades técnicas del sistema.
- **Tipo de compresión**: con pérdidas o sin pérdidas.

Entre los códecs más utilizados en servicios de telefonía y VoIP encontramos los G.711, G.722, G.723, G.726, G.729 y AMR, la mayoría especificados por la UIT-T.

6.1.4.2 G.711

Es un estándar UIT-T liberado en 1972 que utiliza PCM para la codificación: muestras de 8 bits para señales de voz, grabadas a una velocidad de 8000 muestras/segundo generando por tanto una corriente de 64 Kbps.

Existen 2 algoritmos principales que definen el estándar: algoritmo mu-law (usado en América del Norte y Japón) y algoritmo a-law (usado en Europa y el resto del mundo). Ambos son

logarítmicos, pero el segundo fue específicamente diseñado para ser más fácilmente procesado por una computadora.

Este códec a menudo es descrito como un descompresor que utiliza el mismo ratio de muestreo de la telefonía tradicional. Sin embargo, usa una gran cantidad de ancho de banda para la transmisión por lo que no es usado normalmente, aunque puede ser aceptado en entornos LAN (Red de área local) como por ejemplo, telefonía IP en redes de 100 Mbps.

6.1.4.3 G.722

Es un estándar UIT-T de 7 KHz que funciona a 48, 56 y 64 Kbps. Fue aprobado por la UIT-T en noviembre de 1988 y se basa en una codificación ADPCM por sub-bandas.

Se trata de una evolución natural del conocido G.711, cuya calidad es mejor por su ancho de banda de voz más amplia de 50-7000 Hz. Está pensado para RDSI (Red Digital de Servicios Integrados) y canales de 64 Kbps.

6.1.4.4 G.723

Es un estándar UIT-T de códec de voz que describe un algoritmo de bajo ratio de compresión. Comenzó como una extensión del G.721 pero resultó obsoleto tras la liberación del G.726.

El estándar explica dos versiones: una para 5,3 Kbps y otra para 6,4 Kbps. En general, ofrece bajo ancho de banda para la transmisión de la voz y es particularmente adecuado para transmitir voz sobre IP en conexiones WAN (Red de área amplia) de bajo ancho de banda.

6.1.4.5 G.726

Es un estándar UIT-T para voz liberado en 1984 y basado en tecnología ADPCM que opera a velocidades de 16-40 Kbps. El modo más utilizado frecuentemente es 32 Kbps, ya que es la mitad de la velocidad del G.711, aumentando la capacidad de utilización de la red en un 100%.

6.1.4.6 G.729

Se trata de otro estándar UIT-T para voz de 8 KHz que comprime el audio en segmentos de 10 ms. Opera a 8 Kbps usando CS-ACELP (predicción lineal de código algebraico excitado en estructura conjugada), pero existen extensiones de 6.4 Kbps y 11.8 Kbps.

El anexo B de G.729 ofrece un módulo VAD (detección de actividad vocal), otro para mejorar los parámetros del sonido de ambiente en entornos ruidosos y un generador de CNG (ruido de confort), porque en un canal de comunicación, si la transmisión se detiene porque no hay actividad vocal, el otro extremo puede asumir que la transmisión ha sido cortada.

6.1.4.7 AMR

El códec AMR (Multi-tasa adaptativo de las siglas en Inglés Adaptive Multi-Rate) es un formato de compresión de audio para voz adoptado como el estándar de codificación de audio por

3GPP (*3rd Generation Partnership Project*) [69] en Octubre de 1998. Inicialmente fue desarrollado únicamente para GSM (Sistema global para las comunicaciones móviles) pero en 1999 fue estandarizado por el ETSI (Instituto Europeo de Normas de Telecomunicación) [70], y en 2006 fue incluido en la especificación CableLabs PacketCable 2.0 [71].

Se trata de un códec obligatorio para la voz de banda estrecha y otros servicios en redes móviles [72] con tecnologías como WCDMA (Acceso múltiple por división de código de banda ancha), EDGE (Tasas de datos mejoradas para la evolución de GSM) o GPRS (Servicio general de paquetes vía radio).

Existen básicamente dos tipos de códec AMR:

- **AMR-NB**: opera en banda estrecha.
- **AMR-WB**: opera en banda ancha (16 KHz) y fue estandarizado en la recomendación ITU-T G.722.2. Toma 14 bits por cada muestra y tiene 9 posibles velocidades de codificación, entre 6.6 y 23.85 Kbps.

Pero en términos generales y en este proyecto, AMR se refiere únicamente a AMR-NB.

6.1.4.7.1 Modos de Funcionamiento

AMR proporciona flexibilidad en cuanto al ancho de banda que utiliza ya que al ser un codificador adaptable en banda estrecha (200-3400 Hz), es capaz de operar a 8 tasas de bits variable (modos) en función de las condiciones del canal y de la red. En realidad, hay un total de 14 modos, de los cuales 8 están a FR (*full rate channel*) y el resto en HR (*half rate channel*).

Las diferentes tasas de bits se basan en muestreos a 8 KHz con ventanas de 20 milisegundos, lo que da como resultado 160 muestras por ventana.

En la tabla siguiente se muestran los principales modos de funcionamiento de AMR:

Modo	Tasa de bits (kbps)	Canal	Tamaño ventana (Bytes)
AMR_12.20	12.20	FR	32
AMR_10.20	10.20	FR	27
AMR_7.95	7.95	FR/HR	21
AMR_7.40	7.40	FR/HR	20
AMR_6.70	6.70	FR/HR	18
AMR_5.90	5.90	FR/HR	16
AMR_5.15	5.15	FR/HR	14
AMR_4.75	4.75	FR/HR	13
AMR_SID	1.80	FR/HR	

Tabla 46. Modos de funcionamiento de AMR.

Algunos de estos modos son equivalentes a los códecs de voz utilizado actualmente en los sistemas de comunicación móvil. Por ejemplo, el códec AMR de 12,2 Kbps (modo 8) es igual al codificador de GSM EFR (*Enhanced Full Rate*).

Considerando una conversación telefónica normal donde los participantes se alternan en la comunicación, se puede considerar que cada sentido de la transmisión ocupa aproximadamente un 50% del tiempo en media. Por tanto, se tiene una actividad discontinua que el códec AMR debe tratar eficientemente mediante tres funciones básicas:

- Se establece un módulo VAD en el transmisor.
- Se procede a evaluar el ruido acústico de fondo también en el transmisor, generando una serie de parámetros que caracterizan a dicho ruido.
- Los anteriores parámetros como información del ruido de confort se encapsulan en una trama SID (Descriptor de inserción de silencios), que se envía hasta el receptor en intervalos regulares de tiempo.
- Se genera el ruido de confort en el receptor en periodos en los que no se reciben tramas de voz normales.

6.1.4.7.2 Proceso de codificación

AMR es un código algebraico resultado de un algoritmo ACELP (predicción lineal de código algebraico), cuyo diagrama de bloques se muestra a continuación:

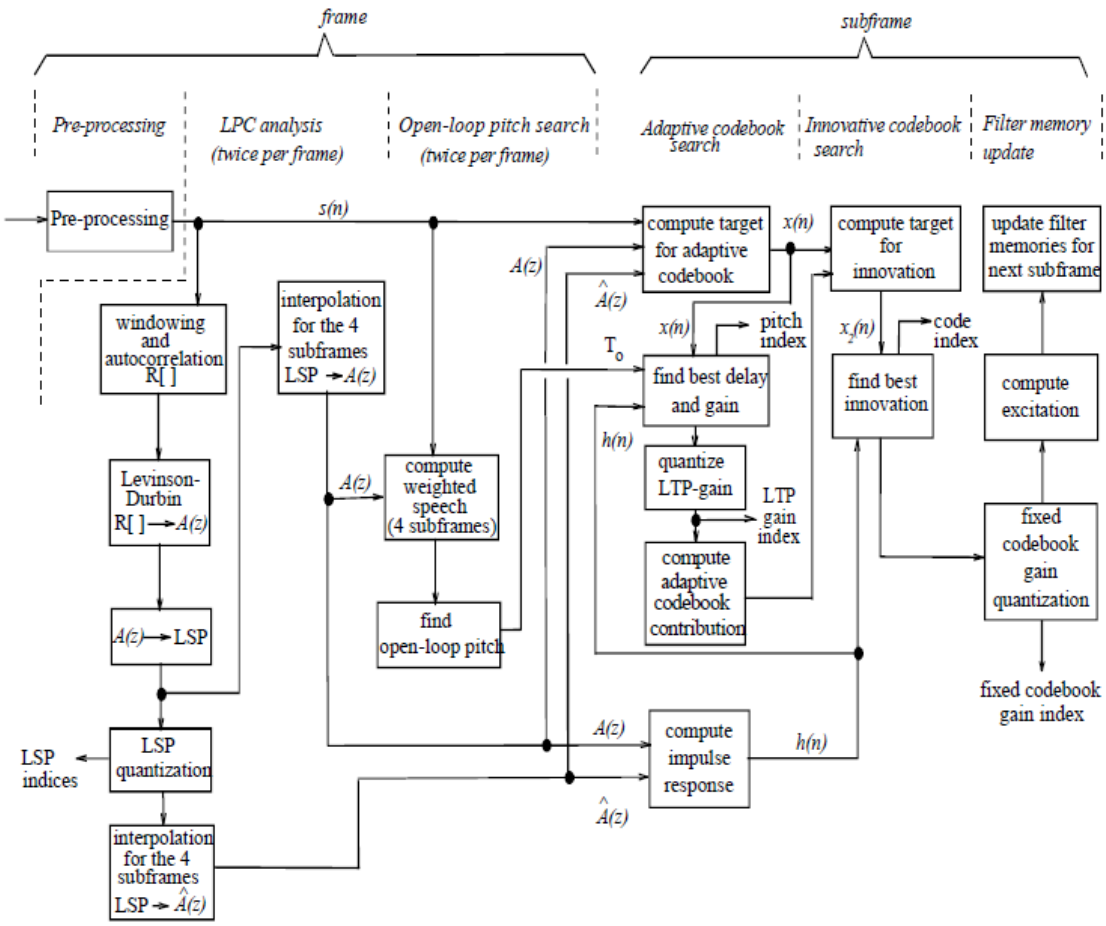

Figura 23. Diagrama de bloques de codificación de AMR.

Antes del proceso de la codificación se aplican dos funciones de pre-procesamiento:

1. Filtrado paso-alto: para filtrar ruido de parpadeo y también interferencias a 50/60 Hz producidas por la voz humana.
2. Reducción de escala: para reducir al mínimo las probabilidades de desbordamiento durante el procesado de operaciones de punto fijo.

A continuación, cada ventana de 20 ms es dividida en 4 sub-ventanas, con 40 muestras y 5 ms de duración cada una. A su vez, por cada ventana se extraen los parámetros LP o coeficientes de los filtros LP del modelo CELP. Finalmente, se pueden obtener tramas de diferente tamaño en función de la cuantificación de los parámetros LP, es decir, del número de bits utilizados para cada parámetro.

Independientemente del tamaño de las tramas obtenidas, a la salida de codificador los bits se reordenan en función de su importancia subjetiva y su sensibilidad frente a errores en tres clases: clase A, Clase B y Clase C. La clase A es la que dispone de los bits más sensibles y, por tanto, es la que necesita también un canal más robusto para la transmisión.

6.1.4.7.3 Estructura del formato

Los ficheros con este formato [73] constan de una cabecera de 6 bytes que lo identifica como fichero de audio codificado en AMR (0x23, 0x21, 0x41, 0X4D, 0x52, 0x0A) siendo una característica unívoca y común para todos los ficheros de este tipo.

A partir del 7º byte, el fichero AMR consta de una serie de ventanas con datos. Cada ventana equivale a 20 milisegundos de audio y contiene 95, 103, 118, 134, 148, 159, 204 o 244 bits en función del modo de funcionamiento usado respectivamente (ver tabla 3).

Dado que cada ventana se puede codificar utilizando cualquiera de los 8 modos válidos de AMR, existe una cabecera de 1 byte por cada tipo de ventana. Dentro de esta cabecera, los bits del 2º al 5º más significativos representan el CMR (*Codec Mode Request*), un valor entre 0-7 que identifica el modo. El primer bit de la cabecera en realidad puede ser ignorado, a pesar de que se utiliza en el caso de necesitar transmitir audio codificado en AMR en tiempo real. Por último, los 3 bits menos significativos de la cabecera son reservados y no se utilizan.

6.1.4.7.4 Ventajas

Algunas ventajas de AMR en relación al resto de códecs de voz mencionados anteriormente corresponden a las siguientes:

- Es el único códec de banda estrecha que ofrece ocho diferentes tasas de bits que pueden ser adaptadas de acuerdo a la congestión de la red, permitiendo así la mejora significativa de la calidad de servicio.
- Hace uso de tecnologías como VAD y CNG.
- La funcionalidad de DTX (Transmisión discontinua) está incluida resultando imprescindible en conversaciones de voz.
- Actualmente el códec AMR está soportado por casi cualquier teléfono móvil en el mundo, incluyendo millones de nuevos dispositivos cada año.

- En rigurosas pruebas comparativas de diferentes códecs para voz dentro de los proceso de normalización llevados a cabo por la ETSI y 3GPP, AMR ha superado todos los requisitos de robustez a la pérdida de paquetes, errores de bit y ruido de fondo, además de demostrar un rendimiento consistente en varios idiomas, incluyendo Inglés (Estados Unidos y Reino Unido), Francés, Alemán, Mandarín, Italiano, y Español.

6.1.4.7.5 Aplicaciones

Las aplicaciones más comunes donde se hace uso de este códec son las siguientes:

- Servicios multimedia para sistemas de comunicaciones móviles 3G.
- Voz sobre IP.
- Telefonía WIFI.
- Audio y Video Conferencias.
- Dispositivos de audio portátiles.
- Mensajería unificada.
- Radio Digital por *Broadcasting*.
- Streaming de audio.
- Herramientas de creación y descarga de contenidos.

6.1.5 Contenedores

Un contenedor es un tipo de formato de archivo capaz de almacenar información de vídeo, audio, subtítulos, y otros metadatos de acuerdo a una especificación concreta.

A pesar de que la lista de contenedores para voz es muy amplia, se ha considerado que los tres formatos más comunes son MP4, 3GP y WAV.

6.1.5.1 MP4

MP4 es un formato de archivo contenedor que forma parte del estándar MPEG-4 (*Moving Pictures Experts Group*) parte 14. Se utiliza para distribuir vídeo y audio pero también puede almacenar otro tipo de datos como subtítulos, información de capítulos e imágenes fijas.

La extensión asociada a los archivos que cumplen este estándar es .MP4, aunque también se asocia a la extensión .M4A, adoptada por Apple, o a las extensiones .M4V o .MP4V.

Habitualmente transporta el códec H.264 AVC (Códec de Vídeo Avanzado) de alta compresión para vídeo o el formato AAC (Codificador avanzado de audio) para audio, aunque también admite MP3 (Formato de audio del estándar MPEG-1) o AMR.

6.1.5.2 3GP

3GP es otro formato contenedor para teléfonos móviles para almacenar información de audio y video, con especificaciones que abarcan las redes GSM, incluyendo GPRS y EDGE y WCDMA.

Fue creado por 3GPP y tiene su origen en la *ISO 14496-1 Media Format*, que a su vez es similar al formato de Quicktime o QTFF.

En cuanto a los códecs que habitualmente transporta:

- El video es almacenado como MPEG-4 o H.263.
- El audio es almacenado en los formatos AMR-NB o AAC-LC (baja complejidad).

6.1.5.3 WAV

WAV (Formato de audio Waveform) es un contenedor de audio digital desarrollado y propiedad de IBM y Microsoft para almacenar audio en los ordenadores personales con calidad de CD.

Entre sus características más importantes destaca la posibilidad de disponer de archivos mono y estéreo a diversas resoluciones y velocidades de muestreo. Por otro lado, el tamaño de los archivos resultantes tiende a ser excesivamente grande al ser un estándar que no emplea compresión alguna, razón por la cual nunca ha sido popular en Internet, donde son más habituales los formatos comprimidos con pérdida.

Es recomendado para uso profesional al no tener pérdida de calidad, siendo PCM el formato de audio más habitual que transporta.

6.1.6 Justificación del códec de voz elegido

La codificación de voz depende absolutamente del sistema operativo móvil y de sus APIs disponibles. Para el estudio de viabilidad de este proyecto, han sido elegidas dos APIs nativas de Android, cada una de las cuales permite la grabación de audio en un formato diferente:

6.1.6.1 API para audio PCM

Esta API es capaz de manejar audio sin comprimir en formato PCM con varias configuraciones. Aunque no permite el uso de otros códecs posibilita elegir la frecuencia de muestreo y el número de canales para PCM por ejemplo.

6.1.6.2 API para audio AMR

Esta API soporta el tratamiento de audio con compresión mediante el códec AMR-NB. Además permite, entre otras muchas opciones, elegir la tasa de codificación en bits por segundo y el uso de otros códecs como AMR-WB o diversos tipos de ACC.

Sin embargo, se ha elegido únicamente el códec AMR-NB debido a la calidad de la voz que ofrece en relación a la compresión obtenida y al hecho de que es ampliamente usado para codificar la voz en las redes móviles actuales. Se han descartado el resto de códecs ya que sólo se soportan en las últimas versiones de Android.

Por tanto, se han elegido los formatos PCM y AMR para codificar la voz que envía StreamDroid.

6.2 Streaming

6.2.1 Introducción

Streaming de multimedia es una tecnología que permite la distribución de audio y vídeo a través de Internet y reproducir o pre-visualizar archivos en tiempo real o por demanda, esto es, sin tener que hacer una descarga previa completa del archivo antes, tratando de minimizar el tiempo de espera.

El medio de transmisión o canal para transmitir información multimedia se comparte con infinidades de aplicaciones que hacen uso del internet por lo que se debe codificar y comprimir dicha información de tal manera que no afecte al resto de comunicaciones y se pueda tener un uso eficiente del ancho de banda.

La figura representada a continuación explica de una forma muy sencilla el funcionamiento del streaming para el caso de una petición web:

Figura 24. Ejemplo de streaming de contenidos web.

Básicamente, el usuario selecciona un archivo multimedia desde un dispositivo o medio y se envía una petición al servidor web. Posteriormente el servidor web re-direcciona la petición al servidor de streaming, el cual envía una respuesta directa al cliente en forma de flujo continuo de datos. El cliente dispone de un reproductor, el cual decodifica los datos y finalmente reproduce el archivo.

Esta tecnología es un requisito para este proyecto, ya que permite la recepción de la voz del profesor por el servidor de subtitulado en tiempo real. A continuación, se realiza una breve explicación de los protocolos de comunicación más comunes en transmisiones multimedia así como de las herramientas más extendidas para reproducir contenidos.

6.2.2 Protocolos de red orientados a Streaming

La principal prioridad para transmitir audio y video en tiempo real es recibir los datos tan pronto como sea posible para obtener un flujo continuo. Sin embargo, en este caso, la integridad completa de los datos no es esencial, por lo que se permite tener pérdidas de algunos paquetes de datos.

HTTP (Protocolo de transferencia de Hiper-Texto), FTP (Protocolo de Transferencia de Archivos), y todos aquellos que funcionan sobre TCP, son protocolos en los que se garantiza la entrega de todos los datos y en un orden específico; es decir, prima la exactitud de la información y no la velocidad.

Por ello, generalmente se evita usar estos protocolos para hacer streaming y se opta por los basados en UDP, que no garantiza la entrega ni el orden de entrega, pero que son más rápidos, lo cual resulta idóneo teniendo en cuenta la latencia que existe en las redes móviles.

6.2.2.1 UDP

UDP (Protocolo de datagramas de usuario) [74] es un protocolo del nivel de transporte basado en el intercambio de datagramas sin que sea necesario previamente establecer una conexión, ya que en la cabecera del propio datagrama se incorpora suficiente información de direccionamiento. Tampoco tiene confirmación ni control de flujo, es decir, los paquetes pueden adelantarse unos a otros, y tampoco se sabe si ha llegado correctamente pues no hay confirmación de entrega o recepción. Cualquier tipo de garantía para la transmisión de la información debe ser implementada en capas superiores.

UDP es generalmente el protocolo utilizado en la transmisión de vídeo y voz a través de una red porque no hay tiempo para enviar de nuevo paquetes perdidos cuando se debe reproducir voz por ejemplo en tiempo real.

En la figura siguiente se muestra la cabecera UDP, la cual consta de 4 campos de los cuales 2 son opcionales: puerto de origen y el checksum.

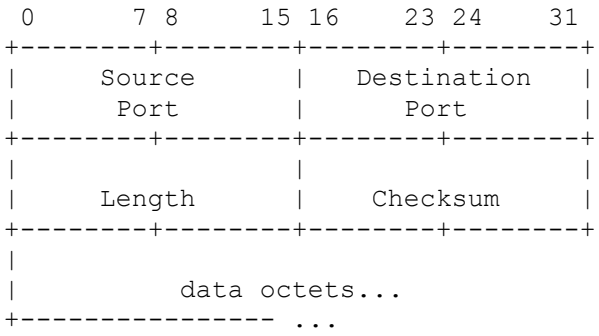

Figura 25. Cabecera UDP.

El puerto origen es opcional ya que el emisor nunca solicita respuestas. En caso de no ser utilizado, el puerto origen debe ser puesto a cero. A los campos del puerto destino le sigue un campo obligatorio que indica el tamaño en bytes del datagrama UDP incluidos los datos y otro campo opcional para el checksum.

6.2.2.2 RTP

RTP (Protocolo de Transporte de Tiempo Real) [75] es un protocolo de nivel de sesión que proporciona entrega de datos en tiempo real independientemente del protocolo subyacente.

Es desarrollado por el grupo IETF (Grupo de Trabajo de Ingeniería de Internet), y se publicó por primera vez como estándar en 1996 con el RFC 1889 y actualizado en el 2003 con el RFC 3550.

RTP permite identificar el tipo de datos a transmitir, obtener el orden para la presentación de los paquetes y sincronizar los flujos de diferentes fuentes.

No hay garantía en el orden de entrega, de hecho, no está garantizado que se reciban todos los paquetes. El receptor debe reconstruir la secuencia enviada por el servidor y detectar las pérdidas de datos.

Por último, se pueden establecer sesiones RTP entre un grupo de aplicaciones que pretendan comunicarse mediante este protocolo. Dicha sesión se identifica por una dirección de red y un par de puertos, donde uno se usa para transmitir los datos y el otro se usa para el control.

La siguiente figura muestra la cabecera utilizada por el protocolo RTP:

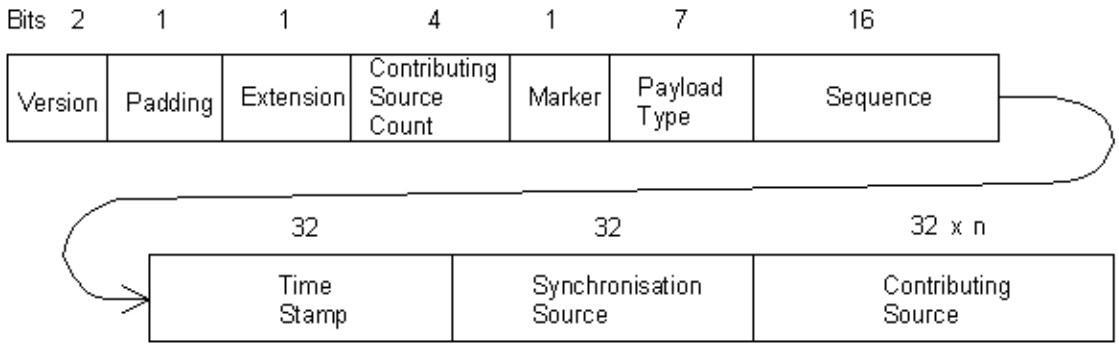

Figura 26. Cabecera RTP.

- **Versión** (2 bit): Los primeros dos bits identifican la versión del protocolo.
- **Padding** (1 bit): El bit del relleno indica que hay uno o más bytes al final del paquete que no son parte de la carga útil. El último byte en el mensaje UDP indica el tamaño del relleno. El relleno es usado por algunos algoritmos de encriptación.
- **Extensión** (1 bit): El bit de extensión se usa para indicar que el encabezado fijo es seguido por una extensión del encabezado. Este mecanismo de la extensión permite implementaciones para añadir información al encabezado RTP.
- **Contributing Source Count CSRC** (4 bits): Número de "fuentes contribuyentes" incluidas en la cabecera de RTP. Si la cuenta CSRC es cero, entonces la fuente de sincronización es la fuente de la carga útil.
- **Marker** (1 bit): Este bit es utilizado para indicar el *frame*. Por ejemplo, puede indicar el inicio de una conversación en RTP: el primer frame.
- **PayLoad type** (7 bits): Un índice en una tabla de perfiles multimedia que describe el formato de carga útil. Los mapeos de carga útil para audio y video están especificados en el RFC 1890 [76].

- **Sequence Number** (16 bits): Número de Secuencia. Un único número de paquete que identifica la posición de este en la secuencia de paquetes. El número del paquete es incrementado en uno para cada paquete enviado. Se utiliza para permitir al receptor de detectar paquetes perdidos o que lleguen en desorden
- **Timestamp** (32 bits): Refleja el instante de muestreo del primer byte en la carga útil. Varios paquetes consecutivos pueden tener el mismo.
- **Synchronization Source SSRC** (32 bits)**:** El identificador de fuente de sincronización (SSRC) es un número de 32 bits que identifica de manera única una sola fuente en un flujo RTP. En una conferencia multimedia, cada emisor escoge un SSRC aleatorio por ejemplo.
- **Contributing Source CSR** (32 bits)**:** El identificador de fuente contribuyente (CSR) es utilizado sólo cuando varios flujos RTP pasan a través de un mezclador. Si hay más de 15 fuentes contribuyentes, sólo 15 pueden ser identificadas.

Dado que RTP no dispone de mecanismos para asegurar la entrega a tiempo u otras garantías de calidad del servicio, existe un protocolo de control (RTCP) para monitorizar la calidad de la distribución de datos.

6.2.2.3 RTCP

RTCP (Protocolo de control de RTP) es un protocolo de comunicación a nivel de aplicación que genera información de control en una transmisión de datos multimedia en tiempo real. Opera en el transporte y gestión de paquetes junto con RTP, pero no transporta ningún tipo de dato multimedia, únicamente de control o de naturaleza estadística.

Cada participante en la sesión RTP recibirá periódicamente paquetes RTCP. Estos paquetes RTCP pueden contener información sobre QoS (calidad del servicio), información sobre la fuente del contenido que se está transmitiendo y estadísticas sobre los datos que se están transmitiendo como bytes enviados, paquetes enviados, paquetes perdidos o variación del retardo durante la transmisión, es decir, el *jitter*.

El objetivo final, por tanto, de RTCP es incrementar la QoS lo máximo posible, ya sea limitando el flujo con algún procedimiento o fijando un códec de compresión más baja o alta en función de las necesidades.

Existen los siguientes tipos de paquetes RTCP:

- **Informes de emisor**. El emisor activo de una sesión envía informes sobre estadísticas de transmisión y recepción.
- **Informes de receptor**. El receptor activo de una sesión puede igualmente enviar estadísticas de la recepción.
- **Descripción de la fuente**. Contiene los CNAMEs y otros datos que describen la información de los emisores como el nombre real del usuario y su e-mail. Estos son utilizados en la interface del usuario para permitir identificar las personas.
- **Paquetes de control específicos de la aplicación**. Varios paquetes RTCP pueden ser enviados en un mismo mensaje UDP.

6.2.2.4 RTSP

RTSP (Protocolo de flujo de datos en tiempo real) [77] se usa para controlar y establecer uno o varios flujos sincronizados de datos multimedia. El control incluye posicionamiento en el flujo, grabación e incluso control del dispositivo.

Es un protocolo no orientado a conexión que no especifica el protocolo de transmisión que debe usarse para los datos, lo que le convierte en una posibilidad muy flexible y extensible. Habitualmente se usa TCP para datos de control del reproductor la mayor parte de las veces y UDP para los datos multimedia aunque también puede usarse TCP si la situación lo requiere. En la siguiente figura se ilustra dicha situación:

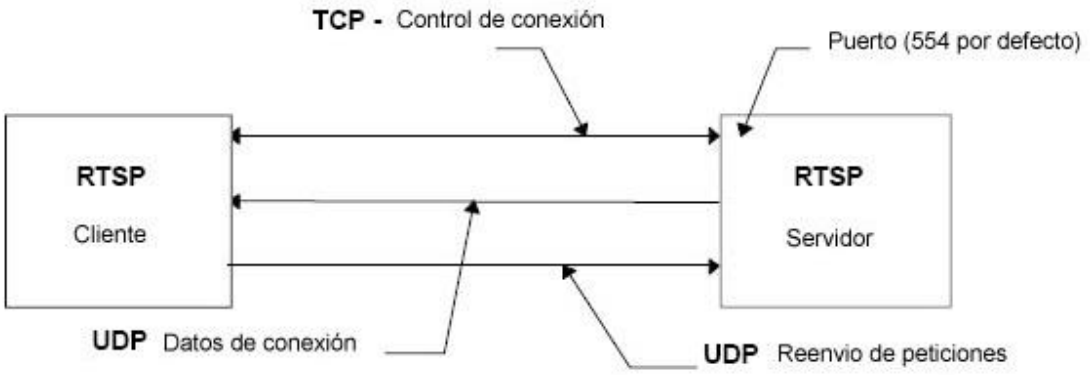

Figura 27. Protocolos de transporte usados en una transmisión con RTSP.

En todo momento el servidor RTSP mantiene un identificador de sesión asociada a una transmisión multimedia y su estado.

El protocolo es muy similar en sintaxis y operación a HTTP por lo que resulta fácil de leer, entender y depurar. Sin embargo, RTSP difiere de HTTP en ciertos aspectos:

- Tanto el servidor como el cliente RTSP pueden lanzar peticiones.
- RTSP presenta el concepto de sesión como parte de la definición del protocolo.
- La petición URI (Identificador de recursos uniforme) siempre contiene la URI absoluta.
- RTSP presenta nuevos métodos y difiere en el identificador de protocolo.
- El protocolo para el transporte de datos no suele ser el mismo.

Las peticiones RTSP suelen enviarse desde el cliente al servidor. A continuación se describen las más habituales:

- **DESCRIBE**: Este método obtiene una descripción de un recurso multimedia almacenado en un servidor. Constituye la fase de inicialización del RTSP.
- **SETUP**: Especifica una URL (Localizador Uniforme de recursos) y un puerto para los datos RTP y otro para RTCP. Cada flujo de datos debe ser configurado con SETUP.
- **PLAY**: El servidor comienza a enviar usando la información útil de SETUP.
- **PAUSE**: Detiene temporalmente uno o todos los flujos. Se puede reanudar con PLAY.
- **TEARDOWN**: Detiene definitivamente la entrega de datos y además se liberan los recursos asociados.

6.2.3 Herramientas de Reproducción de Streaming

Los datos multimedia se pueden reproducir en una amplia variedad de aplicaciones, ya sean reproductores multimedia o páginas Web que incluyan módulos o plugins de estos reproductores. A continuación se realiza una breve introducción a cuatro de los reproductores más comunes de contenido multimedia por streaming.

6.2.3.1 FFmpeg

Figura 28. Logotipo de FFmpeg.

FFmpeg [78] es una solución software multiplataforma creada para grabar, convertir y emitir audio y vídeo que se distribuye en función de las opciones de configuración elegidas bajo licencia LGPL (*Lesser General Public License*) o GPL [79]. Surgió como una iniciativa de Fabrice Bellard y fue desarrollado bajo GNU aunque puede ser compilado bajo otros sistemas operativos como Microsoft Windows o MAC OS X.

FFmpeg dispone de una lista extensa de formatos de fichero, ya sea de vídeo, audio, imagen o subtítulos, así como de una enorme variedad de códecs. Esto es posible gracias a las librerías de las que dispone, en especial *libavformat* y *libavcodec* y a que sus creadores dejan abierta la posibilidad de usar librerías externas para extender aún más su funcionalidad.

De hecho, la librería *libavcodec* es líder en códecs de audio y vídeo, y actualmente es usada por una gran cantidad de proyectos externos a FFmpeg [80], entre los que destacan los siguientes:

- **Google Chrome**: el navegador web de Google.
- **MPlayer**: reproductor de películas multiplataforma.
- **VLC**: reproductor multimedia descrito en esta misma sección.
- **xine**: reproductor multimedia gratuito.

Por otro lado, los principales componentes que constituyen FFmpeg son los siguientes:

- **ffmpeg**: herramienta de línea de comandos para convertir audio o video de un formato a otro. También puede capturar y codificar audio y video en tiempo real.
- **ffserver**: servidor de streaming de emisiones en directo que soporta RTP, RTSP y HTTP.
- **ffplay**: reproductor multimedia basado en SDL (*Simple DirectMedia Layer*) y las bibliotecas FFmpeg.
- **libavcodec**: biblioteca que contiene todos los códecs de FFmpeg. Muchos fueron desarrollados desde cero para asegurar una mayor eficiencia y un código reutilizable.
- **libavformat**: biblioteca que contiene los multiplexadores/demultiplexadores para los principales contenedores de video y audio.
- **libavutil**: biblioteca de apoyo que contiene todas las rutinas comunes.
- **libpostproc**: biblioteca con funciones de post-procesado de vídeo.
- **libswscale**: biblioteca de escalado de vídeo.

6.2.3.2 Quicktime

Figura 29. Logotipo de Quicktime.

QuickTime [81] es un framework multimedia estándar diseñado por Apple que incluye un reproductor multimedia y transmisor de contenidos de alta calidad en Internet denominado *QuickTime Player y* un conjunto de bibliotecas. La primera versión de QuickTime fue lanzada el 2 de Diciembre de 1991.

A partir de su última versión, la 7, Apple ha decidido incorporar nuevas tecnologías como la norma H.264/MPEG-4 AVC, la cual permite tratar contenidos muy nítidos superiores al estándar de DVD y otros formatos de alta calidad y ofrece imágenes en alta definición nítidas y brillantes usando menos ancho de banda y espacio de almacenamiento.

Existe una versión Pro que añade diversas funcionalidades como la edición de vídeo, codificación a variados formatos o la posibilidad de grabar audio o video con un micrófono.

Actualmente se encuentra disponible para los sistemas operativos Windows y Mac OS X. Los sistemas GNU/Linux pueden usar QuickTime mediante programas como MPlayer.

6.2.3.3 RealPlayer

Figura 30. Logotipo de RealPlayer.

RealPlayer [82] es un reproductor multiplataforma desarrollado por *RealNetworks* [83] siendo compatible con numerosos formatos contenedores incluyendo MP3, MP4, formato QuickTime, Windows Media, y los formatos *RealAudio* y *RealVideo* propietarios. También está disponible para muchos sistemas operativos como Windows, Linux, Palm OS, Android, y Mac OS X.

La primera versión fue presentada en Abril de 1995 como *RealAudio Player* y fue uno de los primeros reproductores de medios capaces de realizar streaming a través de Internet. De hecho, la versión 4.01 se incluyo como seleccionable para la instalación de Windows 98.

En mayo de 2015 se anunció desde la web oficial que RealPlayer se había sustituido por *RealTimes*, un nuevo producto que realiza montajes multimedia a partir de fotografías y vídeos de los usuarios, crea copias de seguridad y es accesible a través de la nube.

RealPlayer fue adoptado inicialmente por muchos usuarios como una buena alternativa web a la hora de reproducir streaming de audio y video, pero desde principios del siglo 21, tecnologías como Adobe Flash y HTML5 han sustituido a RealPlayer para este propósito.

6.2.3.4 VLC

Figura 31. Logotipo de VLC.

VLC media player [84] es un reproductor y framework multimedia de código abierto distribuido bajo la licencia GPL2 con soporte para los siguientes sistemas operativos: Microsoft Windows, GNU/Linux, Mac OS X, Solaris, Windows CE y otros sistemas derivados de BSD.

Se inició como un proyecto académico en 1996 por estudiantes de la École Centrale Paris, los cuales diseñaron un sistema para transmitir videos en la red del campus universitario. Actualmente es desarrollado a nivel mundial y coordinado por la organización VideoLAN [85].

VLC incluye de forma nativa un gran número de códecs libres, lo que le convierte en uno de los reproductores más versátiles que existen, dado que es capaz de leer una gran cantidad de formatos de audio, vídeo y subtítulos evitando la necesidad de instalar codecs propietarios. Un ejemplo es la biblioteca libavcodec del proyecto FFmpeg, que ya ha sido descrita.

VLC ofrece también soluciones de streaming (ver figura 32) actuando como:

- **Servidor** (unicast y multicast en IPv4 o IPv6): Archivos MPEG-1, MPEG-2 y MPEG-4, DVD, desde una cámara digital de vídeo o vídeos en directo en la red.
- **Cliente**: recibir, decodificar y representar flujos MPEG, ya sean emitidos por la salida de emisión de VLC o por otra aplicación.

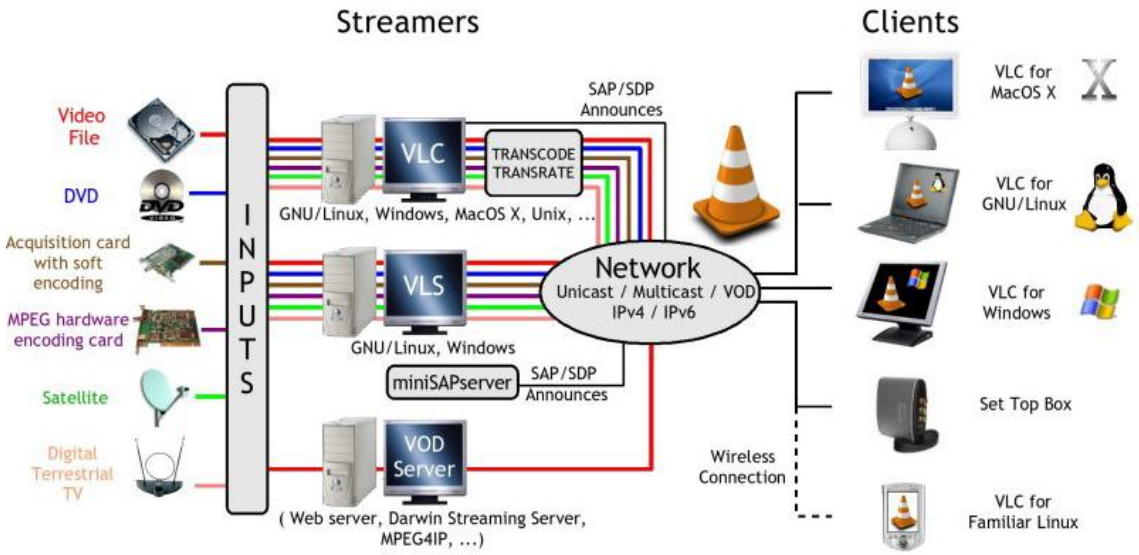

Figura 32. Streaming con VLC.

Por último, VLC dispone de una potente y compleja línea de comandos desde la que se puede replicar cualquier acción disponible mediante la Interfaz gráfica y ampliar aún más las posibilidades que ésta ofrece. Dada la cantidad de información a disposición del usuario acerca del uso de esta línea de comandos, VLC resulta muy útil para tareas y operaciones avanzadas de tratamiento de audio y video. Si se desea obtener una información más completa sobre los comandos de VLC, se puede encontrar en la wiki de VLC [86].

6.2.4 Justificación de las tecnologías elegidas para el transporte y tratamiento del streaming

En este apartado se realizará una justificación de las tecnologías elegidas finalmente para el transporte y tratamiento posterior del streaming de voz que envía StreamDroid hasta el servidor de subtitulado.

6.2.4.1 Protocolo de transporte

Como se explicó anteriormente, UDP es el protocolo base para cualquier transmisión de datos en streaming y proporciona un servicio no fiable donde los datagramas pueden llegar fuera de orden, duplicados, o se pueden perder sin previo aviso. Aplicaciones sensibles al tiempo, tales como la aplicación móvil de este proyecto, a menudo utilizan UDP porque la pérdida de paquetes es preferible a esperar retrasos en la llegada de los paquetes.

La posibilidad de uso de cada uno de los protocolos de transmisión en la aplicación desarrollada está limitada de nuevo por el sistema operativo móvil elegido:

- UDP es el único protocolo de todos ellos que está soportado de forma nativa por Android desde las primeras versiones, por tanto es la primera solución elegida.
- RTP se ha decidido también incluirlo como opción de transporte para el audio por su uso tan extendido en sistemas multimedia pero únicamente hay soporte nativo a partir de Android 3.1. Para solventar este problema, se ha incluido una librería externa que implementa el uso de RTP a través de otras APIs distintas a las nativas.

6.2.4.2 Tratamiento del streaming

Se necesita un reproductor multimedia para manejar el tráfico de voz enviado por el terminal móvil a través de Internet. Este software debe ser capaz de:

- Recibir la voz mediante los protocolos UDP o RTP a través de un puerto de red.
- Reproducción del audio codificado en los formatos elegidos AMR-NB y PCM.

VLC es un reproductor multimedia que soporta múltiples protocolos de comunicación, especialmente para las comunicaciones multimedia, así como múltiples formatos y códecs de audio. También es uno de los reproductores que funciona en un mayor número de plataformas en la actualidad. Además se trata de un software libre.

Después de varias pruebas y comprobar la documentación, VLC ha sido capaz de reproducir el streaming de voz para todas las configuraciones posibles desde un puerto en particular. Los otros reproductores descritos se han descartado por tener menos versatilidad como Quicktime o RealPlayer o por su mayor complejidad para usarlo en el caso de FFmpeg.

En definitiva, los protocolos elegidos para el transporte del flujo de voz son UDP y RTP mientras que VLC es el reproductor multimedia que se utilizará en el servidor de subtitulado.

6.3 Tecnología para el desarrollo del cliente

A continuación se detalla la tecnología final utilizada para el desarrollo de la aplicación móvil de este proyecto, desde una especificación del entorno de desarrollo hasta una especificación de las librerías Android empleadas.

6.3.1 Entorno de desarrollo

Como ya se ha visto en la sección 2.1.2, Android en un sistema operativo para plataformas móviles que se basa en una modificación del núcleo de Linux y que se desarrolla casi en su totalidad por Google.

Google proporciona de forma gratuita un conjunto de bibliotecas desarrolladas o adaptadas para crear aplicaciones para dispositivos móviles Android de cualquier tipo, no sólo smartphones o tablets, al ser una plataforma de código abierto.

Se trata de un Framework de aplicaciones que cuenta con una gran cantidad de APIs disponibles en el lenguaje de programación Java a través de un SDK. Este SDK ha sido precisamente el empleado para desarrollar StreamDroid.

El SDK de Android incluye un conjunto de herramientas de desarrollo como un depurador de código, bibliotecas, un simulador de smartphones, documentación, ejemplos de código y tutoriales. Además, tiene soporte total para todas las versiones de Windows tanto 32 como 64 bits y para otros sistemas operativos.

El IDE utilizado es Eclipse *Standard Kepler*, en cuyo entorno se instala el complemento ADT para permitir el desarrollo de aplicaciones para Android. Este complemento también se proporciona de forma gratuita por Google de la misma manera que el SDK de Android

Eclipse es un software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar aplicaciones de todo tipo. Esta plataforma ha sido típicamente usada para desarrollar entornos de desarrollo integrados, como el IDE de Java, llamado *Java Development Toolkit* (JDT); y el compilador (ECJ), los cuales son parte de Eclipse y a su vez son usados también para desarrollar el mismo Eclipse.

Android Studio, el IDE oficial para el desarrollo de aplicaciones Android desde 2014, ha sido contemplado también como alternativa de entorno de desarrollo, pero la experiencia aportada por el desarrollador de la aplicación móvil de este proyecto desde hace años en Eclipse y la curva de aprendizaje relativamente larga que podría suponer Android Studio por ser un IDE relativamente nuevo, han causado que finalmente fuese descartado.

A lo largo de las siguientes secciones de este capítulo, se detallarán las APIs principales del SDK de Android usadas para la implementación de StreamDroid.

6.3.2 Diagrama de clases

En esta sección se va a describir el diagrama de clases detallado de StreamDroid, con el objetivo de poder disponer de una visión de la aplicación con mayor nivel de detalle de la que se introdujo con el diagrama de bloques funcionales. La siguiente figura ilustra el diagrama:

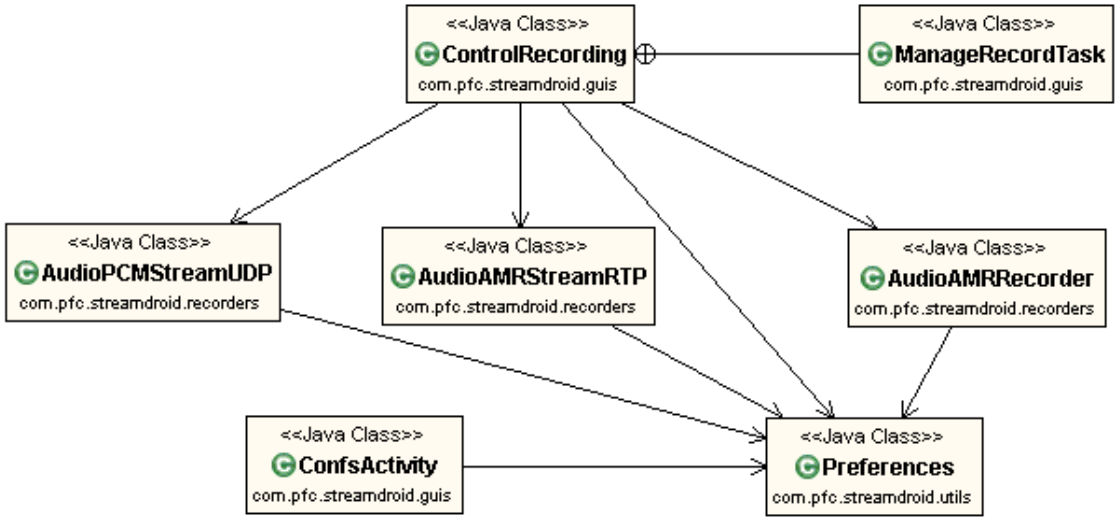

Figura 33. Diagrama de clases para la aplicación StreamDroid

A continuación, se realiza una breve explicación de cada una de estas clases Java:

- **ControlRecording:** Representa la pantalla principal de la aplicación. Contiene por tanto la información del códec de voz y protocolo usado, el temporizador que muestra el estado de la grabación y los dos botones para controlar la grabación. Gestiona también el menú principal de la aplicación, para acceder a la pantalla de configuración o para salir de la aplicación.
- **ManageRecordTask:** Se trata de una tarea asíncrona de Android que depende de *ControlRecording* para gestionar las acciones sobre los botones de control de grabación, por tanto, representa al servicio de gestión de la grabación de voz y envío por streaming.
- **AudioPCMStreamUDP:** Gestiona la configuración, preparación y lanzamiento del servicio que graba audio en PCM y lo envía a través del protocolo UDP por una red de datos en Internet.
- **AudioAMRStreamRTP:** Gestiona la configuración, preparación y lanzamiento del servicio que graba audio en AMR y lo envía a través del protocolo RTP por una red de datos en Internet.
- **AudioAMRRecorder:** Gestiona la configuración, preparación y lanzamiento del servicio que graba audio en AMR y lo almacena de forma local en un fichero.
- **Preferences:** Se trata de una librería que controla todas las variables de configuración de la aplicación y es responsable de la persistencia en general de los datos de configuración.
- **ConfsActivity:** Gestiona a nivel interno el comportamiento de todas las pantallas de configuración de la aplicación.

6.3.3 APIs para el desarrollo de los servicios de grabación de voz

A lo largo del desarrollo de StreamDroid se han usado diferentes APIs nativas de Android para la implementación de los servicios de grabación de voz codificada en un determinado formato y empleando diferentes protocolos de transmisión para su transporte.

6.3.3.1 Grabación de audio PCM y envío sobre UDP

Se hace uso de la clase java nativa y disponible en el API oficial ***AudioRecord***, la cual es capaz de almacenar el audio que se obtiene del micrófono del dispositivo de forma continua, es decir, cada cierto tiempo se guardan los datos a nivel de byte en un buffer.

Los datos que se graban son en realidad flujos de audio en PCM, cuya configuración atiende a los siguientes parámetros:

- Frecuencia de muestreo en Hz. Ejemplos típicos son 44100, 22050, 11025 o 8000. En la aplicación sólo se permite el uso de 8000 o 16000 Hz.
- Configuración de número de canales. Puede ser mono (1 canal) o estéreo (2 canales).
- Nivel de cuantificación. El API permite usar 8 bit o 16 bit por muestra en función de la resolución y de la calidad a obtener. Sin embargo, en la aplicación se permite utilizar sólo PCM-16 bits.

El tamaño del buffer viene determinado por la configuración de estos parámetros y es constante a lo largo de la ejecución de la aplicación.

Por otro lado, el API de Android ofrece clases nativas para la construcción de datagramas que se envían a través de sockets UDP: ***DatagramPacket*** y ***DatagramSocket*** respectivamente.

Una vez que se ha completado el buffer con los datos de audio en PCM, el contenido del mismo se encapsula dentro de un nuevo datagrama que es enviado a través de la red mediante un socket UDP. En la cabecera de este datagrama se especifica el destino y un puerto UDP del mismo para recibir datos. Tras enviar un datagrama, se espera a que el buffer se llene de nuevo con nuevos datos y se crea otro datagrama a partir de ellos, el cual también será enviado por el mismo socket.

Este proceso se repite constantemente hasta que se detiene la grabación de audio, controlado por un objeto de la clase AudioRecord, o hasta que se cierra el socket. En realidad, ambos eventos, están controlados por los botones de control de grabación presentes en la pantalla principal de la aplicación.

Además, se ha diseñado un proceso para calcular una serie de estadísticas acerca del número de datagramas generados, enviados y descartados al finalizar la transmisión.

En resumen, se ha implementado un servicio para Android que es capaz de almacenar el audio que recoge el micrófono del dispositivo en PCM (audio digital sin comprimir) dentro de un pequeño buffer para su posterior envío continuo a través de la red usando para ello el protocolo UDP.

6.3.3.2 Grabación de audio AMR y envío sobre RTP

En función de la versión de Android que use el dispositivo se pueden utilizar uno o dos métodos diferentes para implementar este servicio:

- Clase *MediaRecorder* de la API nativa y librerías de terceros: está disponible para cualquier versión de Android desde la 2.2 y corresponde al diseño inicial.
- APIs nativas para realizar streaming sobre RTP: pueden usarse únicamente a partir de la versión 3.1 de Android y se incorporaron en la implementación de la aplicación más tarde, tras un posterior diseño.

6.3.3.2.1 Clase *MediaRecorder* y librerías de terceros.

MediaRecorder es otra clase java nativa de Android que permite grabar tanto audio como video en varios formatos como 3GP Y MP4 y con diferentes códecs de audio siendo los principales, además de AMR-NB, AMR-WB y AAC. Está especialmente enfocada a la grabación de audio en ficheros pero también puede utilizarse para realizar streaming de voz y video a internet.

Los parámetros de audio que permite esta clase modificar, además del formato y códec de audio, son los siguientes:

- Número de canales: Puede ser mono (1 canal) o estéreo (2 canales). Se ha fijado esta característica a un canal para la aplicación, por lo que no es modificable por el usuario.
- Tasa de bits de codificación: Se permite elegir cualquiera de estas 8 tasas en Kbps: 4750, 5150, 5900, 6700, 7400, 7950, 10200, 12200.
- Frecuencia de muestreo: La velocidad de muestreo depende realmente del formato para la grabación de audio. Por ejemplo, la frecuencia de muestreo para el estándar de codificación AAC es 8-96 KHz, para AMR-NB es 8 KHz, y para AMRWB es 16 KHz. Por tanto, se ha fijado en 8 KHz pues el único códec que se permite para la aplicación usando esta API es AMR-NB.

Para el uso del protocolo RTP en Android, las librerías de software libre de *SipDroid*, un proyecto de un cliente VoIP para Android que ya se detallado en el apartado 3.1, proporcionan dos APIs relacionadas:

- ***RtpPacket***: Permite crear paquetes de datos para transmisiones RTP.
- ***RtpSocket***: Permite crear sockets para comunicaciones sobre RTP.

Además, la clase RtpPacket soporta distintos tamaños para el *payload* o campo de datos del paquete RTP. Esta característica se puede modificar desde una de las pantallas de configuración de la aplicación.

En cuanto al registro del audio, MediaRecorder almacena los datos de audio en buffers intermedios de forma similar a AudioRecord antes de construir el paquete correspondiente**.**

Sin embargo, en esta ocasión si es necesario seguir un estándar definido para el payload de audio codificado en AMR cuando se usa en transmisiones por RTP [87], por lo que ha sido imprescindible para cada paquete RTP:

- Definir la cabecera del payload especificando el primer byte y a continuación los bytes identificadores del tipo de frame, el cual viene dado por la tasa elegida para AMR. El número de frames depende del tamaño del payload seleccionado.
- Especificar el tipo de payload o naturaleza de los datos de audio.
- Crear la estructura correcta del payload a partir del buffer de datos.
- Establecer el *timestamp* y el número de secuencia.

Finalmente se envía el paquete RTP a través del socket definido con el destino y el puerto.

En el siguiente diagrama se pueden observar los distintos estados para el control de grabación:

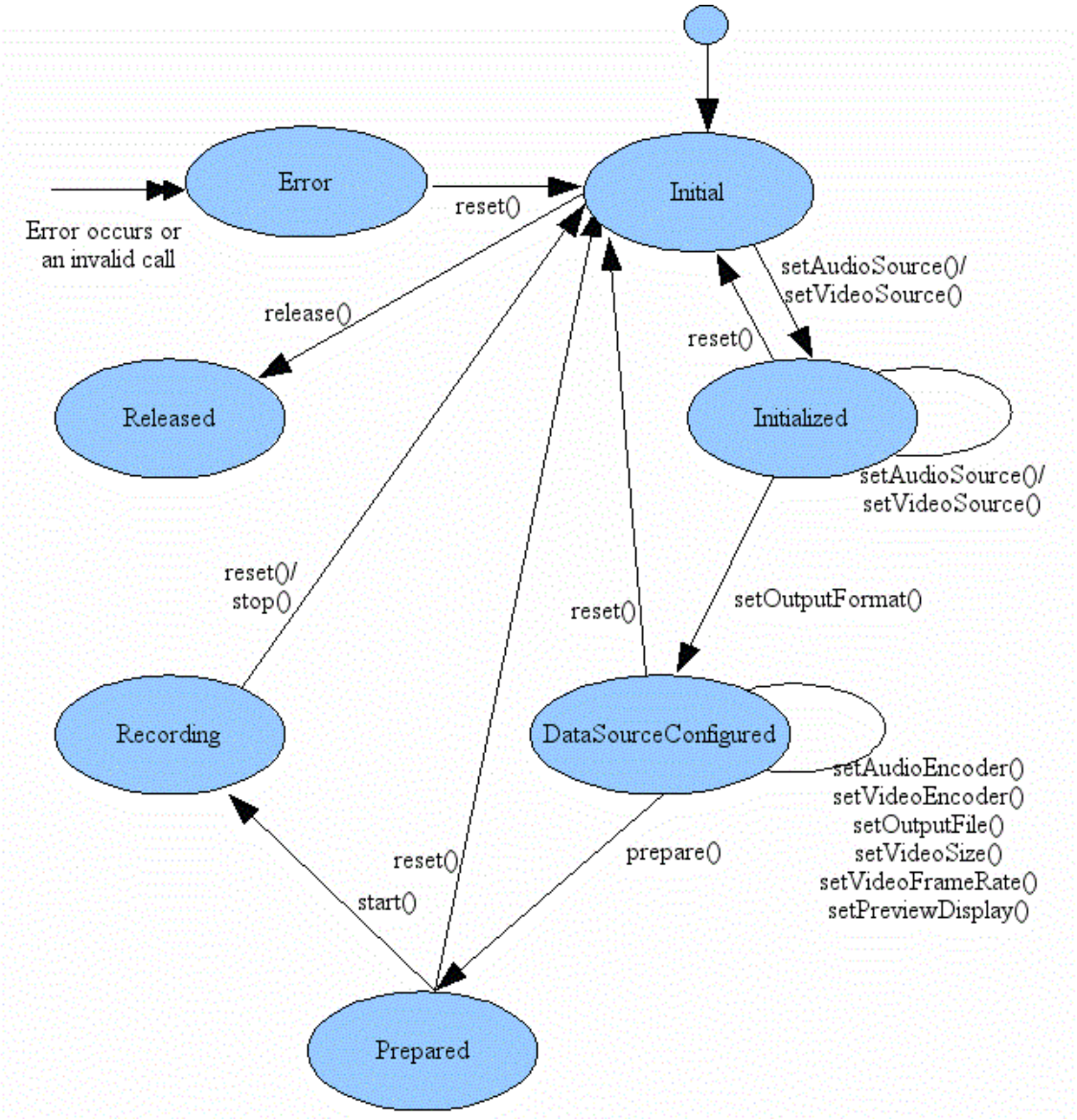

Figura 34. Diagrama de estados de la clase MediaRecorder.

Por tanto, se ha implementado un servicio para Android que es capaz de registrar el audio del micrófono del dispositivo codificado en AMR (audio digital con compresión) y almacenarlo para su posterior envío continuo como si se tratará de un streaming a través de la red usando para ello el protocolo específico RTP.

6.3.3.2.2 APIs nativas para realizar streaming sobre RTP.

El paquete ***android.net.rtp*** del SDK oficial de Android proporciona APIs nativas para RTP, lo que permite a las aplicaciones gestionar en demanda o de forma interactiva un streaming de datos. En particular, las aplicaciones que trabajen con VoIP, por ejemplo en conferencias o para el simple envío de un streaming de audio, pueden utilizar estas APIs para iniciar sesiones y transmitir o recibir flujos de datos a través de cualquier red disponible.

Para ello se pueden usar las siguientes clases:

- ***RtpStream***: Representa la clase base para enviar y recibir paquetes con payloads de datos multimedia sobre RTP.
- ***AudioStream***: Es un flujo que transporta payloads de audio sobre RTP. Además, permite especificar el extremo remoto y gestionar el resto de la configuración de red y de los códecs de audio.
- ***AudioGroup***: Es una clase para gestionar el audio, mezclando todos los *AudioStreams* y, opcionalmente, interactuando con el altavoz y el micrófono del dispositivo al mismo tiempo.
- ***AudioCodec***: Esta clase define una colección de códecs de audio para ser utilizado con *AudioStreams*.

Sólo está disponible este conjunto de librerías a partir de la versión 3.1 de Android. Para versiones anteriores, es imprescindible el uso de las librerías de SipDroid para el soporte RTP.

Para la implementación del servicio de grabación de voz en AMR y envío sobre RTP en la aplicación, se ha llevado a cabo el siguiente proceso:

1. Se crea un grupo de audio con AudioGroup.
2. Se crea un AudioStream con la dirección local del dispositivo.
3. Se especifica el códec de audio para el streaming: AMR a 8 KHz y un canal. También se especifica la tasa de codificación entre las 8 disponibles.
4. Se especifica el servidor remoto y puerto de destino en el AudioStream.
5. Para iniciar la grabación, se añade el AudioStream al grupo de audio creado en el inicio del proceso.

Esta última solución para implementar el servicio resulta mucho más apropiada desde el punto de vista del procesado y la simplicidad del código que usando las librerías de SipDroid junto con la API *MediaRecorder*, pero no está disponible para todo el parque de terminales Android y no puede ser contemplada como única implementación.

6.3.3.3 Grabación de audio AMR y almacenamiento en local

La implementación de este servicio se ha realizado usando la clase *MediaRecorder* también pero en lugar de realizar una configuración de red forzando a usar el transporte de red sobre RTP, tan sólo es necesario especificar el fichero y su ubicación dentro del almacenamiento físico del dispositivo. Por tanto, no se necesitan importar las librerías de SipDroid en este caso, siendo el proceso de implementación bastante más simple. Se permite modificar igualmente la tasa de bits de codificación y la frecuencia de muestreo.

6.3.4 Otras alternativas

A continuación se describen dos alternativas para enviar voz por streaming diferentes a las ya mencionadas y que no han sido implementadas en la aplicación móvil de este proyecto.

6.3.4.1 Librerías de GStreamer

GStreamer [88] es un framework multimedia diseñado para ser multiplataforma y muy conocido para entornos Linux. Se desarrolla en el lenguaje C, usando la biblioteca GObject.

GStreamer permite crear aplicaciones audiovisuales tales como la reproducción de música o realizar tareas más complejas como mezclar audio y vídeo. También provee un SDK específico para Android con un conjunto de APIs para escribir aplicaciones.

Aunque se han encontrado casos en los que se ha conseguido utilizar estas librerías con éxito para Android, esta opción es la menos viable por su dificultad y riesgo.

6.3.4.2 OpenCORE y el motor 2-Way

Dentro de la OHA se eligió a la empresa *PacketVideo* [89] para ser la proveedora del subsistema multimedia de Android *OpenCORE*. Lo más destacado de este framework es:

- Soporte a protocolos de streaming como RTSP y HTTP.
- Modos de funcionamiento del streaming de descarga: simple y progresiva.
- Formatos de ficheros: MP3, MP4, 3GP, AAC, AMR, y WAV.

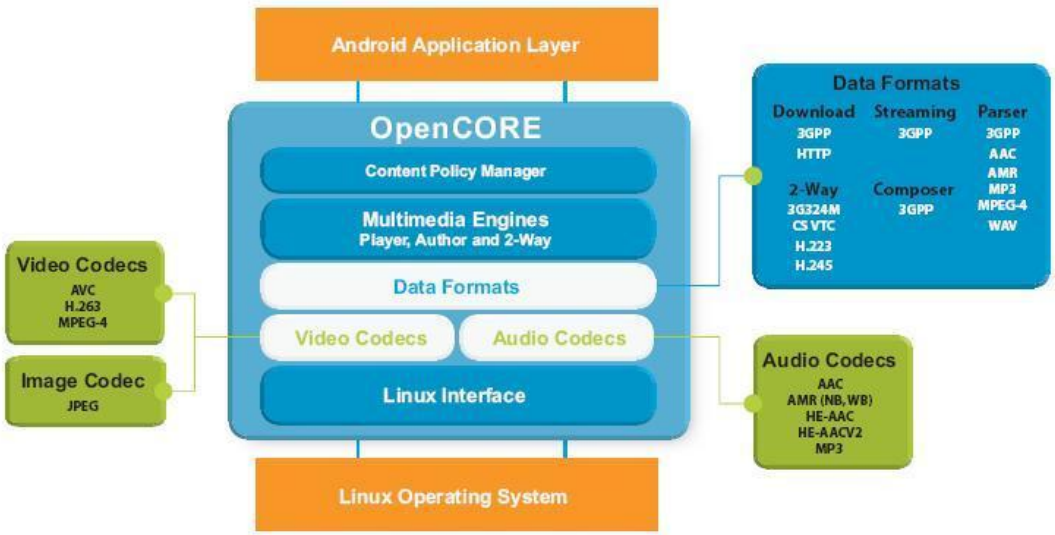

Figura 35. Arquitectura y servicios de OpenCORE

Como se acaba de especificar, OpenCORE está más enfocado a hacer streaming progresivo tanto de descarga como de reproducción. En cualquier caso, proporciona un motor denominado *2-Way* que está pensado para permitir que el flujo del streaming sea tanto de entrada como de salida. Además, este motor permite la posibilidad de hacer video llamadas.

6.4 Tecnología para el desarrollo del servidor

A lo largo de esta sección se detalla la implementación de los elementos básicos software y hardware necesarios en el servidor de subtitulado para disponer del sistema final propuesto para APEINTA al completo.

DNS es el software para reconocimiento automático de habla que se está utilizando en el servidor de subtitulado de APEINTA. La versión que se ha usado es DNS 10.1 Profesional, por lo que para el desarrollo de este proyecto se utilizará la misma.

El principal problema que presenta la versión elegida de DNS para APEINTA es la limitación de las funcionalidades por red, ya que no soporta recibir un streaming de audio como entrada a su sistema de forma directa. La versión server de DNS no presenta este problema y podría procesar directamente el flujo de audio desde la tarjeta de red, pero debido a su alto precio, unos 35000 $, se ha desestimado finalmente.

Sin embargo, se han estudiado otras alternativas para solventar este problema con la actual versión de DNS. Finalmente, una solución viable es el uso un reproductor multimedia y de dos tarjetas de sonido físicas.

6.4.1 Reproductor multimedia

Se necesita un componente software para manejar el tráfico enviado por el terminal móvil a través de internet debido a la limitación por parte de DNS de recibir un flujo de audio directamente por red. Por lo tanto, un reproductor multimedia podría ser un buen componente para llevar a cabo esta funcionalidad.

VLC es el reproductor multimedia elegido para el tratamiento del streaming por las razones que ya se expusieron con anterioridad en este documento.

Es recomendable disponer de la última versión del VLC, la cual se puede obtener a través de su web oficial de forma gratuita. En el momento de realización de este documento, la última versión estable es la 2.2.1, la cual es capaz de reproducir el tráfico de voz UDP o RTP desde un puerto concreto del servidor de subtitulado siendo el contenido audio codificado en PCM o AMR con varias configuraciones diferentes.

Tanto la configuración de VLC para la reproducción de los distintos tipos de flujos de voz descritos, como el entrenamiento de DNS y su posterior utilización para proporcionar un servicio completo de transcripción de voz a texto, pueden consultarse en la Guía de Usuario disponible como Anexo en este documento.

El uso de dos tarjetas de sonido en paralelo para completar la funcionalidad requerida en el servidor, no es la única solución que se ha probado. También se proporcionan más alternativas a esta solución en la Guía de Usuario.

6.4.2 Dos tarjetas de sonido

Es necesario disponer de dos tarjetas de sonido en el mismo servidor de subtitulado ya que la versión de DNS elegida no permite recoger el audio a transcribir directamente de la tarjeta de red. De hecho, la versión usada de DNS únicamente permite especificar como fuente de audio para la transcripción la entrada de una tarjeta de sonido.

Además, las dos tarjetas de sonido necesitan los siguientes requisitos:

- Ambas deben tener una interfaz de entrada/salida para no perder calidad de audio en la transmisión de la señal de voz entre ellas.
- Cable de conexión entre las interfaces SPDIF (Formato de Interfaz Digital Sony/Philips) de cada tarjeta.
- Drivers de cada una de ellas instalados correctamente para que sean reconocidas ampliamente por el Sistema Operativo y también por DNS.

La siguiente figura ilustra en detalle del módulo de conversión de datos y recepción de streaming del servidor de subtitulado incluyendo las dos tarjetas de sonido:

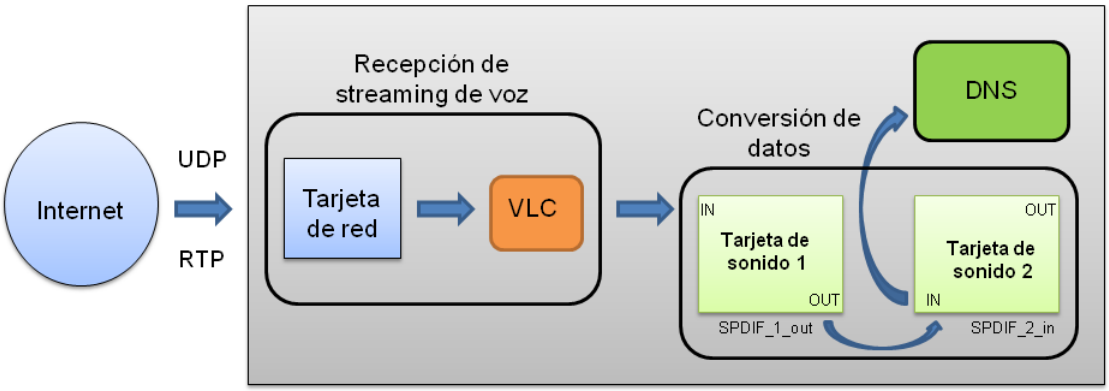

Figura 36. Conversión de datos y recepción de streaming en el servidor de subtitulado

Con la arquitectura propuesta se consigue enviar el flujo de audio que reproduce VLC a la salida de la primera tarjeta (SPDIF_1_out) en tiempo real. Por otro lado, dicha salida (SPDIF_1_out) se conecta con la entrada de la segunda tarjeta (SPDIF_2_in) de forma directa mediante cableado físico, de manera que el audio digital se mapea directamente como fuente de datos para DNS a través de la segunda tarjeta. Es decir, el audio digital se asigna directamente como entrada al servicio de transcripción en tiempo real de APEINTA, por lo que DNS es capaz de reconocer la voz y transformarla en texto para una transcripción o para generar subtítulos.

En resumen, en primer lugar un reproductor de contenido multimedia, como VLC, se necesita para la recepción del flujo UDP/RTP desde la tarjeta de red. Después se necesitan también dos tarjetas de sonido para la conversión de datos y la posterior comunicación directa con DNS para realizar las tareas de subtitulado.

Capítulo 7. Pruebas

Este apartado pretende explicar las pruebas realizadas para demostrar la viabilidad del objetivo principal de este proyecto: transmitir voz en tiempo real desde un dispositivo móvil para la generación de subtítulos en un servidor. Durante el desarrollo del proyecto, se han ido probando los distintos módulos implementados tanto en la aplicación móvil como en el servidor de subtitulado, asegurando con ello la calidad del trabajo realizado y verificando a su vez el funcionamiento correcto del sistema.

Por tanto, el resultado de las pruebas hará referencia tanto a la calidad de la transmisión entre el terminal móvil y el servidor, así como a la eficiencia de DNS para transcribir la voz a texto.

7.1 Consideraciones previas

7.1.1 Dispositivos móviles usados

La elección de Android como plataforma para la aplicación diseñada ya se ha justificado en la sección 2.7. Por tanto, los dispositivos móviles elegidos usarán Android para todas las pruebas.

Dentro del entorno de desarrollo de Eclipse existe un simulador de un dispositivo móvil completo para realizar pruebas: Android Virtual Device (AVD) Manager. Se comenzaron las pruebas usando este emulador pero es extremadamente lento y presenta algunas limitaciones por lo que se decidió finalmente probar con StreamDroid instalado directamente en un terminal móvil real y así comprobar la correcta comunicación e interacción con el servidor.

Por lo tanto, se han realizado las pruebas en dos fases:

- **Fase 1. Smartphone y S.O. antiguo**. Nexus One y Android 2.3.
- **Fase 2. Smartphone y S.O. más moderno.** Galaxy Nexus y Android 4.4.

Se han elegido estos terminales porque han sido diseñados íntegramente por Google y considerados como dos de los dispositivos oficiales de desarrollo para Android, lo que implica:

- El sistema operativo es Android puro, sin ninguna modificación.
- Las actualizaciones de sistema llegan siempre primero a estos dispositivos.
- El sistema está optimizado y adaptado al 100% a estos terminales.
- Precio reducido en comparación a otros smartphones con prestaciones similares.

Ambos permiten configurar opciones de desarrollo con Eclipse simplemente conectando el dispositivo al USB del PC. Con ello, se consigue instalar directamente las aplicaciones en el dispositivo tras ejecutarlas en Eclipse y realizar tareas de depuración complejas.

Ninguno de los 2 terminales elegidos tiene capacidad de conexión a redes 4G, por lo que las pruebas se realizarán con redes 3G o con redes WiFi. En cualquier caso, todas las pruebas que resulten satisfactorias con redes 3G también se podrían extender a redes 4G pues estas últimas ofrecen mayor QoS en todos los aspectos.

7.1.2 Configuración de la codificación de la voz

Se ha elegido una configuración específica de cada tipo de códec usado para comprimir la voz en StreamDroid, PCM o AMR, para todas las pruebas realizadas. La justificación es la siguiente:

PCM:

- Frecuencia de muestreo (8 KHz): es la que se utiliza habitualmente en las señales de voz con calidad telefónica.
- Un solo canal (mono).
- Bits por muestra (16): se ha comprobado que con 8 bits, hay excepciones debido a que en la creación del buffer solo se permiten 16 bits.

Por tanto, teniendo en cuenta que se tienen 16 bits/muestra, 8000 muestras/segundo y un solo canal, la conversión empleará 128 Kbps de ancho de banda.

AMR:

El modo 8 del códec AMR es igual al codificador de GSM EFR, el cual fue seleccionado en 1995 por el ETSI como el códec estándar para GSM. Por tanto, es el tipo de AMR más extendido y usado desde hace muchos años para la codificación de la voz en las redes móviles.

Este códec empleará 12.2 Kbps de ancho de banda ya que solo se emplea un canal.

7.1.3 Análisis de datos recibidos en el servidor

En cuanto al estudio de la recepción del flujo de audio en el servidor de subtitulado, se ha usado *Wireshark* [90], disponible para Windows y más plataformas.

Wireshark, antes conocido como *Ethereal*, es un analizador de protocolos enfocado a realizar análisis y solucionar problemas en redes de comunicaciones, desarrollar software de red y protocolos, y también para funcionar como herramienta didáctica. La funcionalidad que provee es similar a la de *tcpdump*, pero añade una interfaz gráfica y muchas opciones de organización y filtrado de información.

Para analizar el tráfico que envía StreamDroid hasta el servidor, sólo es necesario filtrar en Wireshark por el puerto de escucha y el protocolo de red usado, que siempre será UDP. También es posible codificar el flujo de paquetes filtrando de UDP a RTP cuando corresponda.

La configuración previa para el lanzamiento del VLC se ha realizado con anterioridad a la configuración de la aplicación móvil, de forma que una vez que se decida comenzar la grabación, el servidor se encuentre ya preparado para la recepción del streaming.

7.1.4 Lugar de las pruebas

Todas las pruebas se han realizado en los laboratorios del CESyA durante los meses de Abril y mayo de 2011, a excepción de la última fase de pruebas de Septiembre de 2015, donde no era necesario el servidor de subtitulado de APEINTA.

7.2 Medidas de pérdidas de paquetes

StreamDroid es capaz de calcular una serie de estadísticas acerca de la transmisión cuando se usa el servicio de grabación de audio PCM y envío sobre UDP: el número de datagramas generados y enviados durante el tiempo de transmisión y una lista de aquellos que no han sido enviados o recibidos por cualquier razón, por ejemplo, debido a la pérdida de conexión a través de una red WiFi.

El objetivo de esta prueba es verificar la calidad de la conexión al contar el número de datagramas perdidos, es decir, los datagramas no recibidos en el servidor para dos tipos de redes diferentes: la red de datos 3G y la red WiFi. Esta estadística la proporciona *Wireshark*.

Para cada conexión, el experimento se ha repetido 5 veces (Caso 1...Caso 5), con una transmisión de 60 segundos de duración en cada caso. Los resultados sobre el número de datagramas generados, enviados y recibidos se muestran a continuación.

7.2.1 Red de datos 3G

La transmisión de voz desde el terminal móvil al servidor desde una red de datos 3G se ha probado en dos escenarios diferentes, uno con mala cobertura y otro con buena cobertura.

7.2.1.1 Buena cobertura

Cuando el dispositivo está conectado a una red 3G en un entorno donde existe una gran cobertura, no se detecta la pérdida de datagramas, por lo que es posible hacer una transcripción en tiempo real.

7.2.1.2 Mala cobertura

Cuando el dispositivo está conectado de nuevo a la red 3G, pero en otro entorno cuya cobertura 3G es a veces débil, tampoco se han detectado pérdidas de datagramas, tal y como muestra la siguiente tabla:

	Generados	Enviados	Recibidos	Perdidos (%)
Caso 1	1867	1867	1867	0
Caso 2	1848	1848	1848	0
Caso 3	1861	1861	1861	0
Caso 4	1862	1862	1862	0
Caso 5	1853	1853	1853	0
Media	1858	1858	1858	0

Tabla 47. Estadísticas de pérdida de paquetes en red 3G con mala cobertura.

En este caso, todos los paquetes enviados son recibidos correctamente en el servidor de transcripción obteniendo un 0% de las pérdidas.

Por tanto, el único problema que podría existir al usar redes 3G con mala cobertura sería el retardo. Este aspecto se abordará con detalle en una sección posterior.

7.2.2 Red WiFi

Se ha realizado pruebas cuando el dispositivo está conectado a una red WiFi, ya sea pública (contratada por el uso de varios usuarios, tal como una red inalámbrica pública en la Universidad) o privada (contratada para uso personal).

7.2.2.1 Red WiFi Pública

La red WiFi pública de la Universidad Carlos III de Madrid se ha usado para realizar las pruebas.

La siguiente muestra las estadísticas de la pérdida de datagramas de una red pública WiFi:

	Generados	Enviados	Recibidos	Perdidos (%)
Caso 1	1852	1852	1302	30
Caso 2	1886	1886	1587	16
Caso 3	1893	1893	1699	10
Caso 4	1883	1883	1718	8
Caso 5	1876	1876	1625	13
Media	1878	1878	1586	15,4

Tabla 48. Estadísticas de pérdida de paquetes en una red pública WiFi.

El número considerable en la pérdida de datagramas se debe a la inestabilidad de esta red WiFi, puesto que es necesario establecer una conexión por VPN a las misma dificultando tener disponible una conexión estable. Este hecho provoca cortes en la reproducción de audio en el servidor siendo imposible recuperar esa información. En cualquier caso, estos cortes son generalmente aislados y se producen de forma puntual, con una duración media de 5 segundos.

7.2.2.2 Red WiFi privada

En este caso, no se han detectado cortes en la reproducción de audio ni tampoco pérdida de datos, debido probablemente a que la red WiFi privada es mejor y más fiable que la pública.

En cualquier caso, todos estos resultados no son extrapolables a cualquier caso debido a que el número de pruebas realizadas no es muy grande y al haber utilizado un único escenario, aunque pueden ser tomados como una buena referencia para una primera aproximación.

7.3 Medidas de ancho de banda

El ancho de banda que necesita StreamDroid para enviar los datos en tiempo real a través de una conexión a internet hasta el servidor de subtitulado depende directamente del tipo de codificación empleada para la voz y del tamaño elegido para el campo de datos (payload) de los paquetes transmitidos, independientemente de si se trata de datagramas UDP simplemente o de paquetes RTP.

Por tanto, el ancho de banda demandado es independiente del tipo de red a la que el dispositivo se conecte para realizar las pruebas; es decir, el *throughput requerido* para el envío del audio hasta el servidor será el mismo usando una red WiFi que conectando el dispositivo a una red de datos móvil.

Sin embargo, un tipo de red donde se proporcionen tasas muy bajas de transmisión como una red 2G con GPRS, donde se permite velocidades de transferencia de 56 a 114 Kbps, el audio codificado en PCM no podría transmitirse correctamente pues necesita del orden de 128 Kbps, ligeramente mayor que lo permitido por la tecnología de la red.

El objetivo de esta prueba es conocer el ancho de banda del streaming de audio que se recibe en el servidor en función de distintas configuraciones de códecs de voz y de distintos tamaños de payload. Además, se realizará una comparación con el ancho de banda calculado teóricamente.

Todas las pruebas se realizarán con el dispositivo conectado a una red 3G. También podría haberse utilizado una red WiFi, pero para medir el ancho de banda no es necesario realizar pruebas en ambas redes.

7.3.1 Descripción de las pruebas

Se han enviado 30 segundos de voz desde StreamDroid hasta el servidor con cinco configuraciones diferentes:

- PCM-16bits, 8 KHz, 1 canal.
- AMR 12.2, 8 KHz, 1 canal y tamaño de payload de 50 bytes.
- AMR 12.2, 8 KHz, 1 canal y tamaño de payload de 250 bytes.
- AMR 12.2, 8 KHz, 1 canal y tamaño de payload de 500 bytes.
- AMR 12.2, 8 KHz, 1 canal y tamaño de payload de 1000 bytes.

El tamaño de payload solo puede ser seleccionable para el audio codificado en AMR y si no se activa el uso de las librerías nativas. El tamaño seleccionado por el usuario es sólo aproximado, pues StreamDroid realiza un cálculo interno para ajustar de nuevo el tamaño y en el servidor se reciben los paquetes RTP con este nuevo payload ajustado.

Para cada segundo y para cada configuración, Wireshark ha calculado el ancho de banda aproximado y los resultados se han registrado en la siguiente tabla:

Formato voz	PCM	AMR			
Tamaño payload configurado (bytes)	-	50	250	500	1000
Tamaño payload recibido (bytes)	512	33	225	481	993
Ancho de banda segundo 1	120,859	36,929	16,099	14,272	13,820
Ancho de banda segundo 2	148,378	36,918	16,385	14,437	13,675
Ancho de banda segundo 3	138,547	36,968	16,061	14,395	13,773
Ancho de banda segundo 4	145,954	34,824	16,537	15,207	13,815
Ancho de banda segundo 5	138,639	36,966	16,046	14,431	14,514
Ancho de banda segundo 6	138,002	37,112	16,057	14,376	13,756
Ancho de banda segundo 7	138,374	34,767	16,078	14,397	13,731
Ancho de banda segundo 8	138,496	34,847	16,048	14,386	13,731
Ancho de banda segundo 9	137,501	36,850	15,978	14,399	13,728
Ancho de banda segundo 10	121,776	34,801	17,322	15,253	13,740
Ancho de banda segundo 11	138,234	37,180	16,146	14,396	13,736
Ancho de banda segundo 12	138,470	34,846	16,691	14,370	14,566
Ancho de banda segundo 13	141,099	36,727	16,277	15,227	13,732
Ancho de banda segundo 14	138,236	34,824	16,892	14,474	13,830
Ancho de banda segundo 15	138,347	34,819	16,463	14,391	13,728
Ancho de banda segundo 16	138,434	32,392	16,619	14,430	13,811
Ancho de banda segundo 17	138,583	34,768	16,060	15,353	14,491
Ancho de banda segundo 18	137,873	34,846	16,085	14,397	13,720
Ancho de banda segundo 19	137,770	34,767	16,540	14,466	13,824
Ancho de banda segundo 20	138,467	34,844	16,864	14,623	13,748
Ancho de banda segundo 21	138,323	34,844	15,987	14,391	14,522
Ancho de banda segundo 22	144,012	34,833	15,959	15,273	13,888
Ancho de banda segundo 23	129,596	34,851	15,634	14,390	14,397
Ancho de banda segundo 24	138,018	34,845	16,085	15,070	13,723
Ancho de banda segundo 25	143,265	34,898	16,263	15,205	14,420
Ancho de banda segundo 26	138,322	34,757	15,650	14,342	13,722
Ancho de banda segundo 27	138,370	34,771	15,561	15,412	13,772
Ancho de banda segundo 28	138,725	34,855	16,521	14,541	13,927
Ancho de banda segundo 29	138,224	34,836	16,627	14,411	13,630
Ancho de banda segundo 30	141,444	34,821	16,085	14,444	13,688
Ancho de banda promedio (Kbps)	138,011	35,310	16,254	14,639	13,905
Ancho de banda teórico (Kbps)	138,5	34,238	15,61	14,01	13,28

Tabla 49. Ancho de banda calculado por segundo para audio PCM y AMR.

7.3.2 Cálculo del ancho de banda teórico

Para el cálculo del ancho de banda o throughput teórico se parte de dos variables: tasa de codificación del audio y tamaño de payload.

A partir de estos dos parámetros se puede calcular el número de paquetes de voz generados por segundo:

$$N = \frac{\text{tasa de codificación en bps}}{\text{tamaño payload en bits}}$$

Ahora se puede calcular el tamaño en bits de cada paquete de datos, que será diferente si se trata de un datagrama UDP o de un paquete RTP. Además hay que tener cuenta el tamaño de las cabeceras de los protocolos de red implicados:

- Cabecera RTP: 12 bytes
- Cabecera UDP: 8 bytes
- Cabecera IP: 20 bytes
- Cabecera Ethernet: 14 bytes

S= $((\text{tamaño payload en bytes}) + 8 + 20 + 14)*8$ (para UDP)

S= $((\text{tamaño payload en bytes}) + 12 + 8 + 20 + 14)*8$ (para RTP)

Con el número de paquetes de voz generados por segundo y el tamaño en bits de cada paquete, ya se puede calcular el throughput teórico en Kbps:

$$T = \frac{N * S}{1000}$$

7.3.3 Gráficas de ancho de banda

A continuación se muestran una serie de gráficas para representar la evolución del ancho de banda en Kbps calculado por cada segundo de transmisión y para cada configuración de audio:

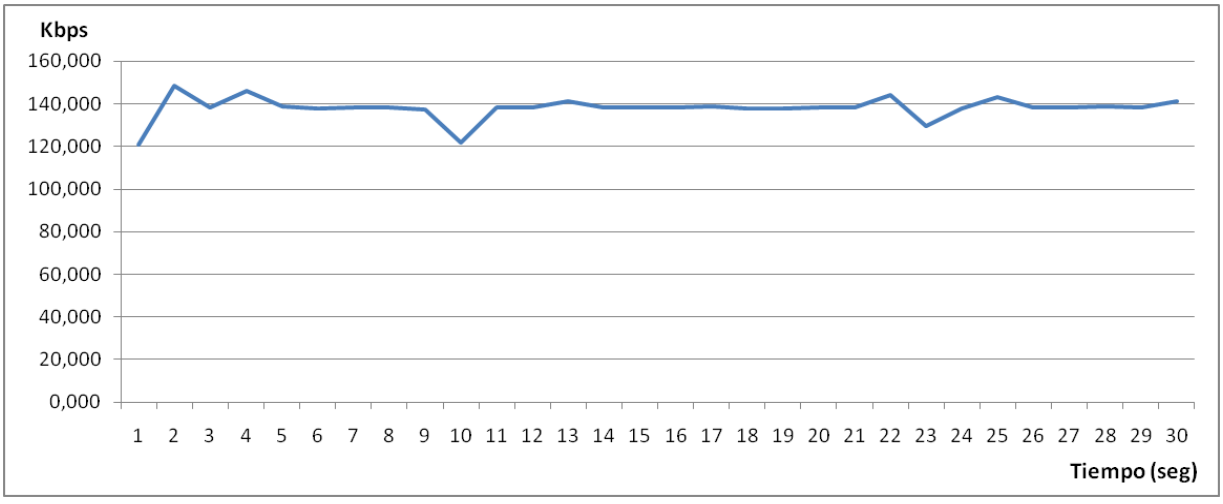

Figura 37. Ancho de banda para audio PCM sin especificar paylaod.

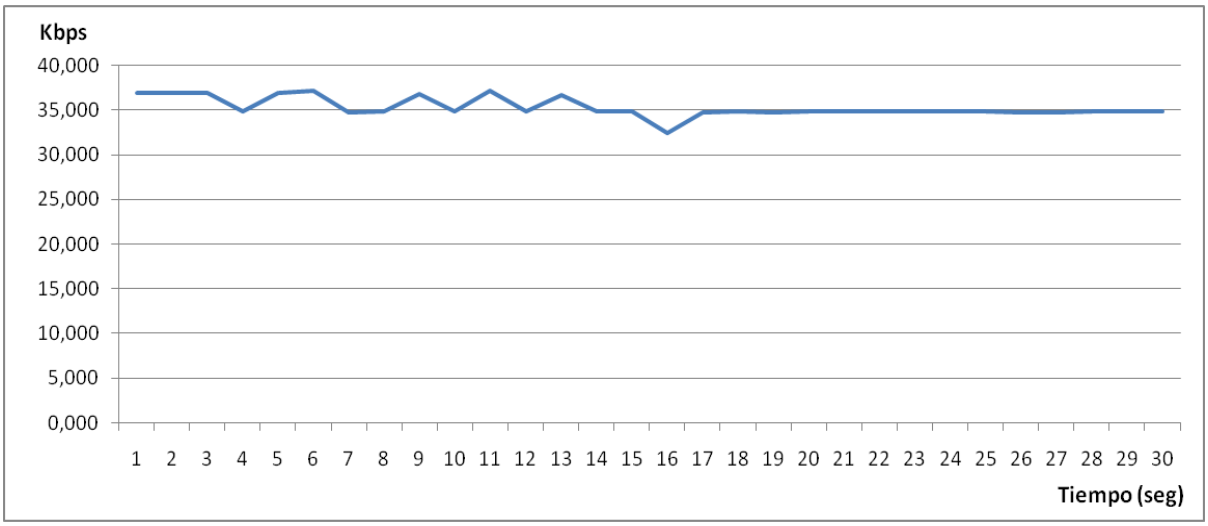

Figura 38. Ancho de banda para audio AMR con paylaod de 50 bytes.

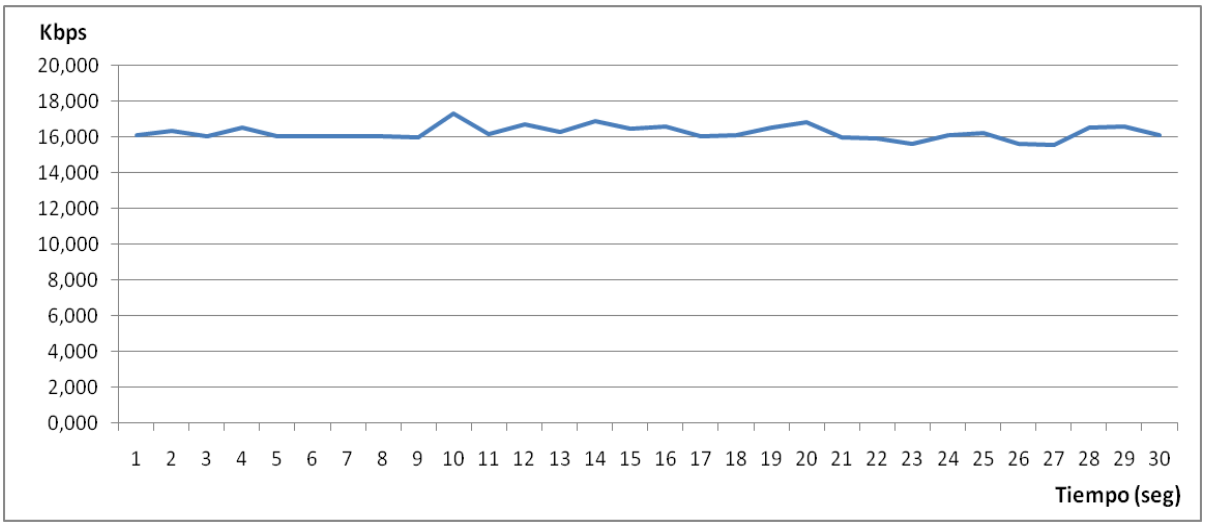

Figura 39. Ancho de banda para audio AMR con paylaod de 250 bytes.

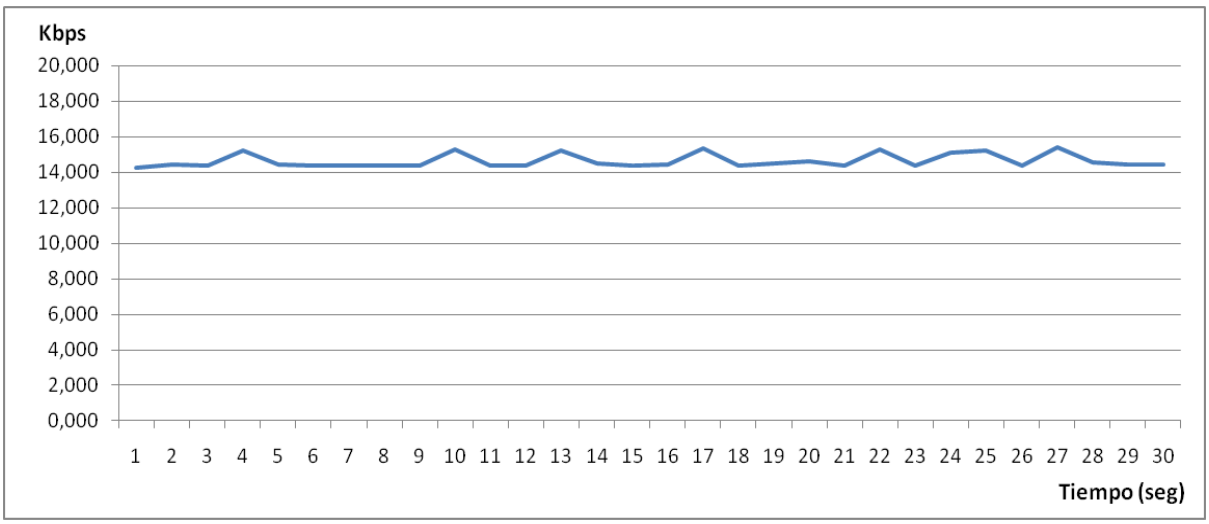

Figura 40. Ancho de banda para audio AMR con paylaod de 500 bytes.

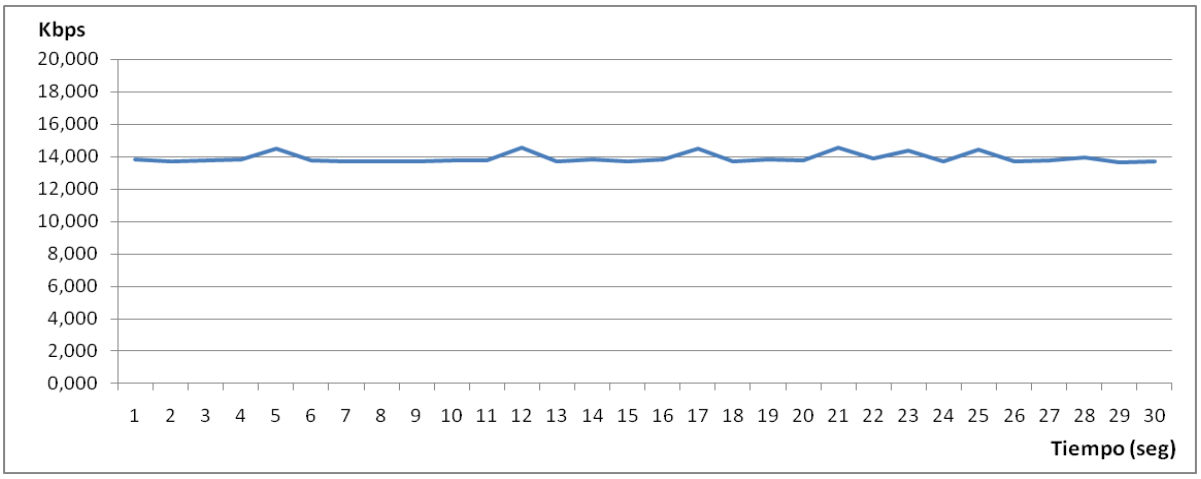

Figura 41. Ancho de banda para audio AMR con paylaod de 1000 bytes.

7.3.4 Conclusiones

En este apartado, se han realizado pruebas para calcular el ancho de banda real durante una transmisión de 30 segundos, extrayendo todos los datos del analizador Wireshark, y también se ha calculado por otro lado el ancho de banda teórico. Todo ello para cinco configuraciones diferentes del tipo de audio y del tamaño de los campos de datos por paquete.

Se recuerda con la siguiente tabla la comparativa entre el ancho de banda real y el teórico:

Formato	PCM	AMR			
Tamaño payload configurado (bytes)	-	50	250	500	1000
Ancho de banda promedio (Kbps)	138,011	35,310	16,254	14,639	13,905
Ancho de banda teórico (Kbps)	138,5	34,238	15,61	14,01	13,28

Tabla 50. Comparativa entre el ancho de banda real y el teórico

Los resultados muestran que los valores entre el ancho de banda promediado de los 30 segundos y el teórico son muy similares. Además, a medida que se va aumentando el tamaño de payload, el ancho de banda real y teórico se aproximan a la tasa de codificación de bits para el tipo de AMR elegido: 12.2 Kbps.

La razón es que cuanto más pequeño es el payload, más sobrecarga sufre el streaming por las cabeceras de los protocolos de red por lo que es más eficiente en términos de ancho de banda transmitir paquetes con payload mayores. Estos protocolos son Ethernet, IP, UDP y RTP.

Por último, es importante destacar que una red 3G es capaz de soportar velocidades de transmisión de datos de hasta 144 Kbps sobre vehículos a gran velocidad, 384 Kbps en espacios abiertos y 7.2 Mbps con baja movilidad (interior de edificios), tasas muy por encima de las experimentadas en estas pruebas.

A la vista de estos resultados, el ancho de banda usado por el streaming de voz no afectará al servicio de subtitulado en el servidor de APEINTA en ningún caso. Sin embargo, el retardo introducido por la red si podría influir negativamente. En la siguiente sección se realizan pruebas de retardo para comprobar este hecho.

7.4 Medidas de retardo

Un bajo retardo en la transmisión de los datos entre el terminal móvil y el servidor resulta un factor decisivo para poder proporcionar un servicio de subtitulado en tiempo real.

Valores de retardo del orden de varios segundos podrían suponer demoras bastantes considerables entre la emisión de la voz en directo y la generación de los subtítulos, dando lugar a un servicio que ya no podría ser considerado de tiempo real.

En esta sección se describirán las pruebas llevadas a cabo para estimar el retardo en la transmisión del flujo de voz que envía StreamDroid con respecto a su recepción en el servidor de subtitulado.

7.4.1 Descripción de las pruebas

Para calcular una aproximación del retardo experimentado, se ha conectado un micrófono al servidor de subtitulado con el objetivo de registrar el ruido ambiente del laboratorio.

Además se configura VLC para que reproduzca por los altavoces del servidor de subtitulado un flujo de audio proveniente del dispositivo móvil en función de las cinco mismas configuraciones diferentes que ya se usaron para las medidas de ancho de banda:

- PCM-16bits, 8 KHz, 1 canal.
- AMR 12.2, 8 KHz, 1 canal y tamaño de payload de 50 bytes.
- AMR 12.2, 8 KHz, 1 canal y tamaño de payload de 250 bytes.
- AMR 12.2, 8 KHz, 1 canal y tamaño de payload de 500 bytes.
- AMR 12.2, 8 KHz, 1 canal y tamaño de payload de 1000 bytes.

Por último, se utiliza el software *Audacity*, una aplicación multiplataforma libre usada para grabación y edición de audio, para grabar todos los sonidos captados por el micrófono.

De esta forma, cuando el usuario emite un sonido seco y agudo (por ejemplo el sonido que produce una palmada), éste queda registrado en el servidor mediante Audacity y a su vez se envía desde el dispositivo móvil hasta el servidor para que se reproduzca gracias al VLC.

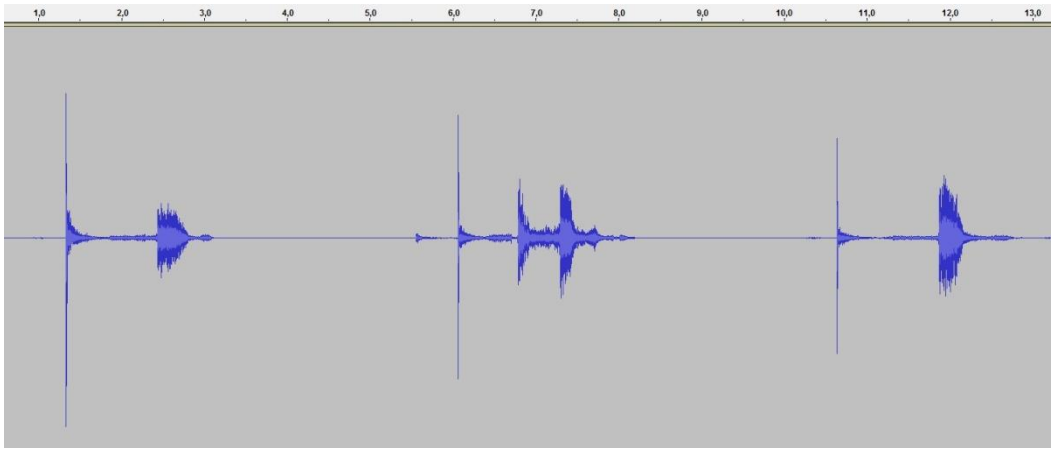

Figura 42. Ejemplo de visualización de alternancia de sonidos y silencios con Audacity

Durante la reproducción del sonido recibido por los altavoces del servidor, éste queda de nuevo registrado por Audacity. Entonces, es posible analizar la diferencia temporal entre ambos, pues Audacity graba el flujo de audio que ha registrado el micrófono de forma continua y a lo largo del tiempo.

Este procedimiento se ha repetido 15 veces para cada una de las configuraciones de codificación de audio mencionadas anteriormente.

El retardo experimentado depende claramente del tipo de red a la que se encuentre conectado el dispositivo móvil en el momento de la realización de las pruebas, por lo que se realizará un estudio para una red de datos 3G y una red WiFi.

7.4.2 Red de datos 3G

A continuación se muestra una tabla que registra los valores de retardo calculados de forma aproximada por el procedimiento descrito anteriormente en una red 3G:

Formato voz	PCM	AMR			
Tamaño payload configurado (bytes)	-	50	250	500	1000
Tamaño payload recibido (bytes)	512	33	225	481	993
Retardo medida 1 (ms)	513	598	507	810	950
Retardo medida 2 (ms)	513	542	503	812	954
Retardo medida 3 (ms)	510	615	502	803	931
Retardo medida 4 (ms)	515	477	503	601	934
Retardo medida 5 (ms)	513	476	504	608	1080
Retardo medida 6 (ms)	513	547	507	598	1010
Retardo medida 7 (ms)	516	568	508	597	1015
Retardo medida 8 (ms)	502	560	513	592	925
Retardo medida 9 (ms)	490	560	516	664	928
Retardo medida 10 (ms)	484	600	512	656	922
Retardo medida 11 (ms)	542	613	548	710	983
Retardo medida 12 (ms)	531	593	547	705	984
Retardo medida 13 (ms)	525	595	550	699	1066
Retardo medida 14 (ms)	542	600	460	751	1065
Retardo medida 15 (ms)	515	580	470	631	1062
Retardo promedio (ms)	515	568	510	682	987
Desviación típica Retardo (ms)	16,06	43,65	25,05	81,20	58,49

Tabla 51. Retardos estimados para audio PCM y AMR en redes 3G

Se ha calculado también el promedio y la desviación típica del retardo para las 15 medidas y por cada configuración de audio.

También se proporciona una gráfica por cada configuración de audio con las 15 pruebas:

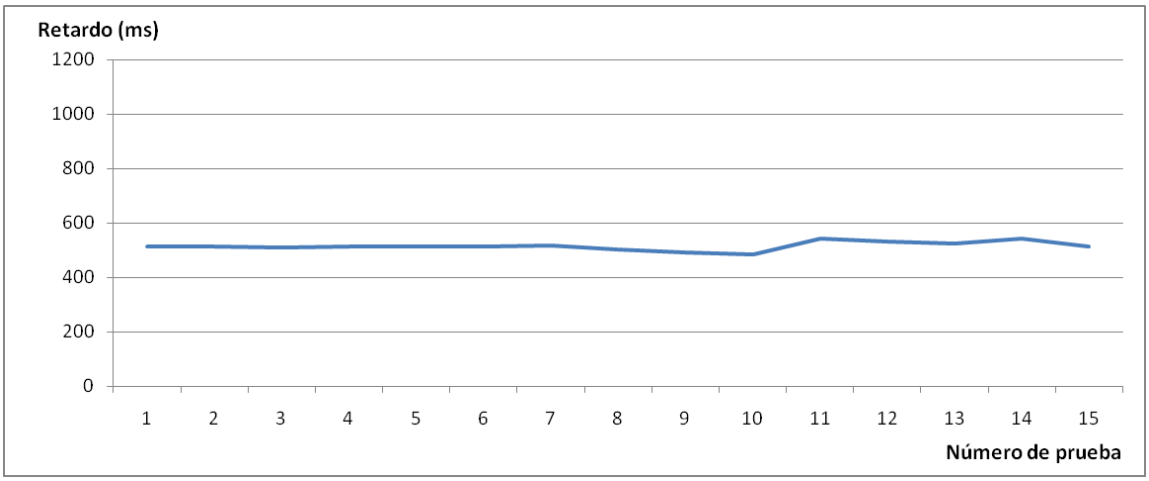

Figura 43. Retardo para audio PCM sin especificar paylaod en red 3G.

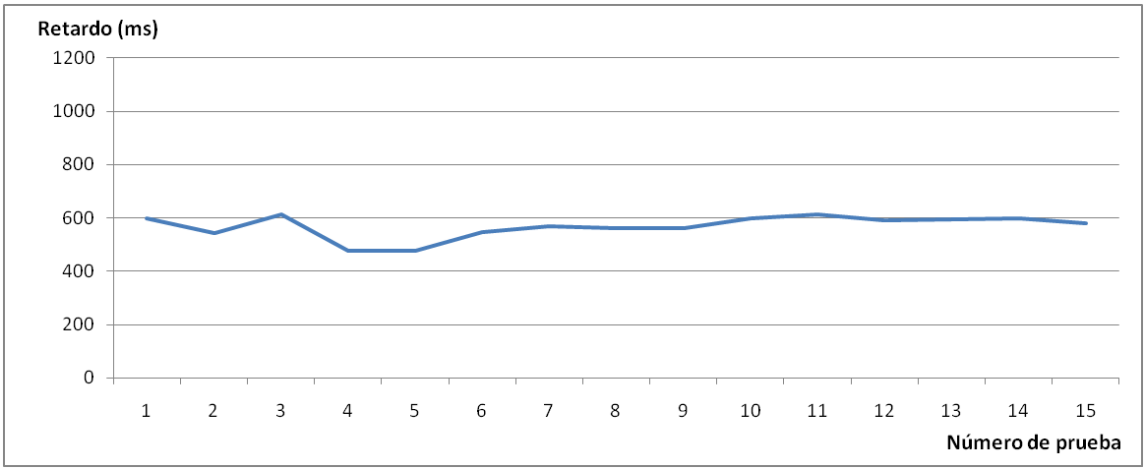

Figura 44. Retardo para audio AMR con paylaod de 50 bytes en red 3G.

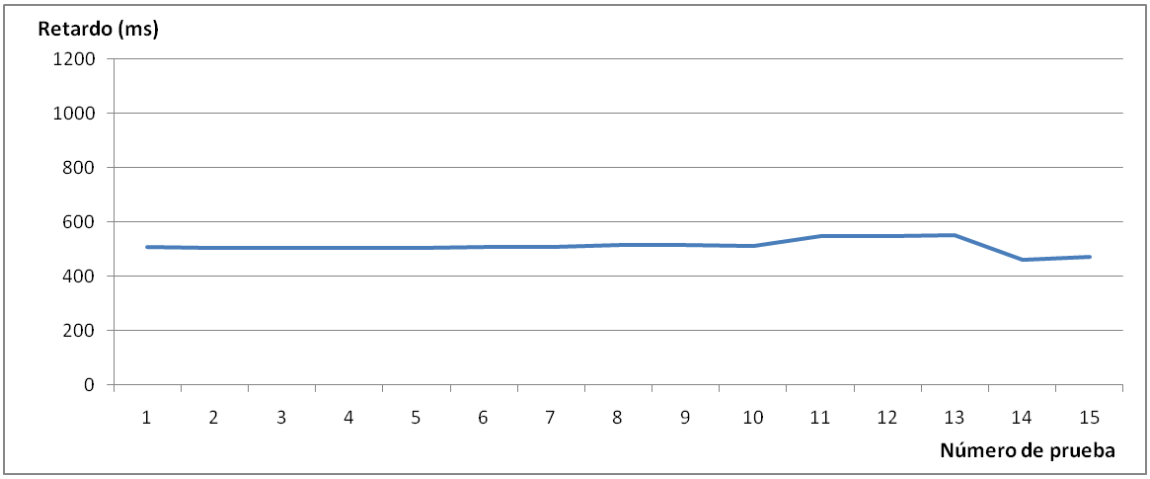

Figura 45. Retardo para audio AMR con paylaod de 250 bytes en red 3G.

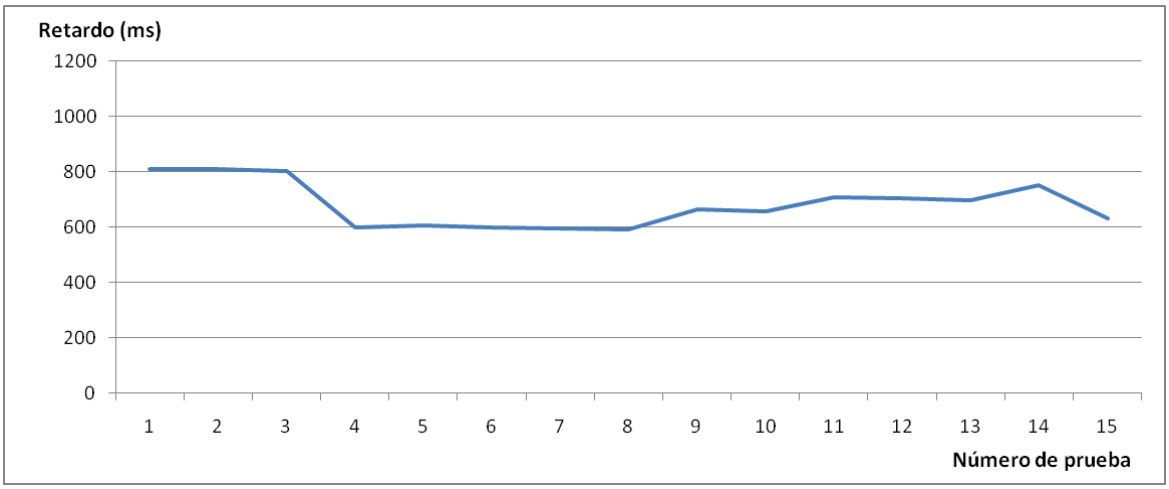

Figura 46. Retardo para audio AMR con paylaod de 500 bytes en red 3G.

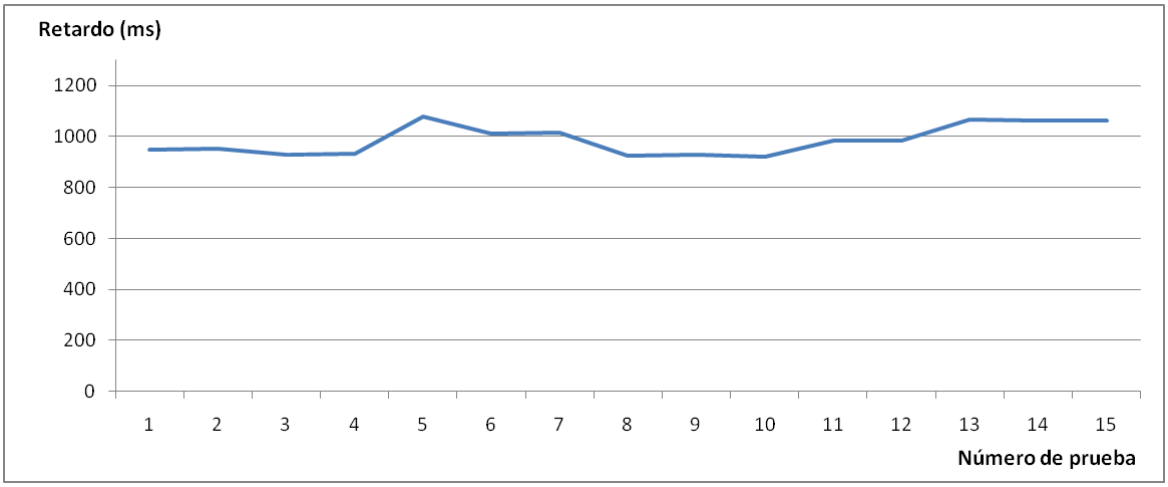

Figura 47. Retardo para audio AMR con paylaod de 1000 bytes en red 3G.

Como conclusión inicial, se puede ver claramente que a medida que se aumenta el tamaño del payload para los casos de uso de audio en AMR, el retardo estimado aumenta así como su variabilidad, lo que puede influir directamente en el aumento del *jitter*.

El jitter se define como una variación en el retardo de los paquetes recibidos. Desde el emisor, los paquetes se envían en un flujo continuo manteniéndose espaciados uniformemente. Debido a la congestión de la red, gestión de colas inadecuadas o errores de configuración, el retardo entre cada paquete puede variar en lugar de mantenerse constante. Para solucionar el problema del jitter, se usan técnicas de bufferización de paquetes.

Por tanto, el códec AMR con un payload de 250 bytes es la opción con mejores prestaciones.

7.4.3 Red WiFi

Cuando el dispositivo se ha conectado a una red WiFi, también se han realizado las mismas medidas que para el caso de una red de datos 3G. En la tabla siguiente se detallan todos los retardos calculados junto con su promedio y desviación típica:

Formato voz	PCM	AMR			
Tamaño payload configurado (bytes)	-	50	250	500	1000
Tamaño payload recibido (bytes)	512	33	225	481	993
Retardo medida 1 (ms)	467	380	394	529	911
Retardo medida 2 (ms)	461	383	391	484	908
Retardo medida 3 (ms)	461	383	380	486	912
Retardo medida 4 (ms)	455	380	380	615	911
Retardo medida 5 (ms)	420	359	435	615	911
Retardo medida 6 (ms)	465	365	412	613	986
Retardo medida 7 (ms)	462	339	429	613	893
Retardo medida 8 (ms)	388	339	429	499	891
Retardo medida 9 (ms)	496	319	426	496	893
Retardo medida 10 (ms)	473	296	429	496	896
Retardo medida 11 (ms)	442	293	429	540	896
Retardo medida 12 (ms)	445	298	429	494	899
Retardo medida 13 (ms)	380	296	423	597	905
Retardo medida 14 (ms)	425	339	435	554	893
Retardo medida 15 (ms)	432	380	432	555	798
Retardo promedio (ms)	445	343	417	546	900
Desviación típica Retardo (ms)	31	35	20	53	37

Tabla 52. Retardos estimados para audio PCM y AMR en una red WiFi

También se han diseñado una gráficas para visualizar más fácilmente los resultados de la tabla anterior. Se muestran a continuación:

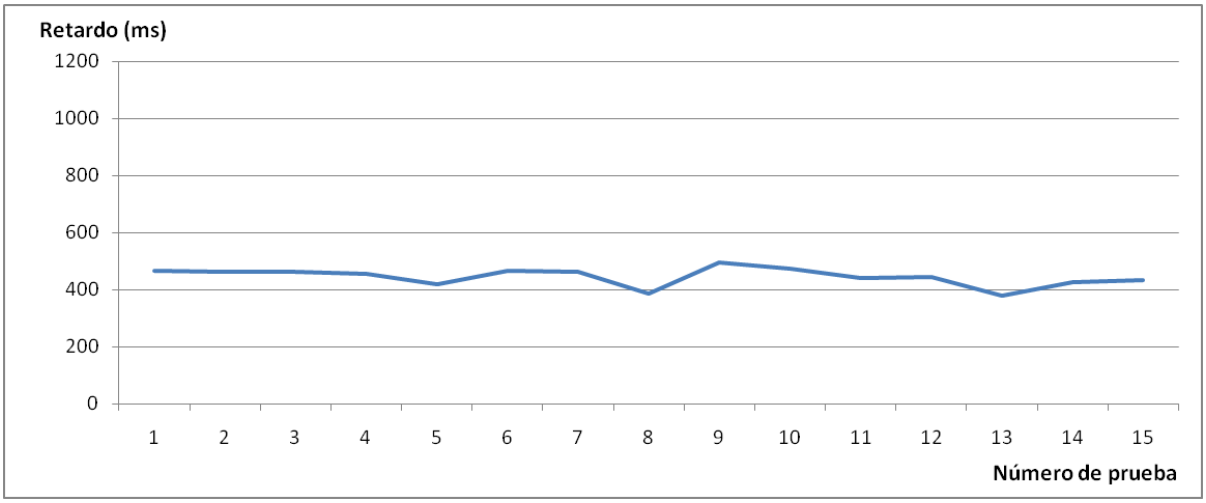

Figura 48. Retardo para audio PCM sin especificar paylaod en red WiFi.

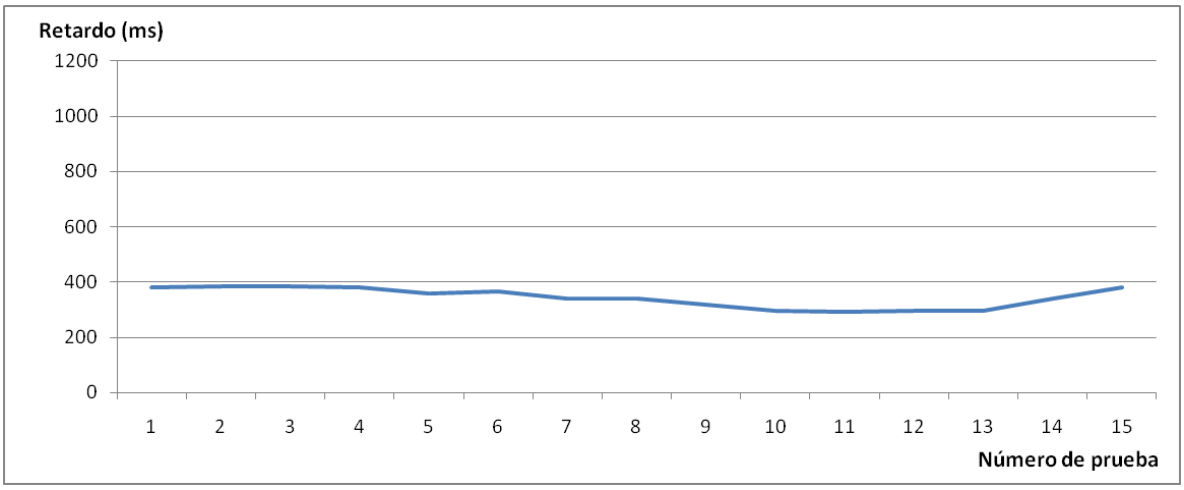

Figura 49. Retardo para audio AMR con paylaod de 50 bytes en red WiFi.

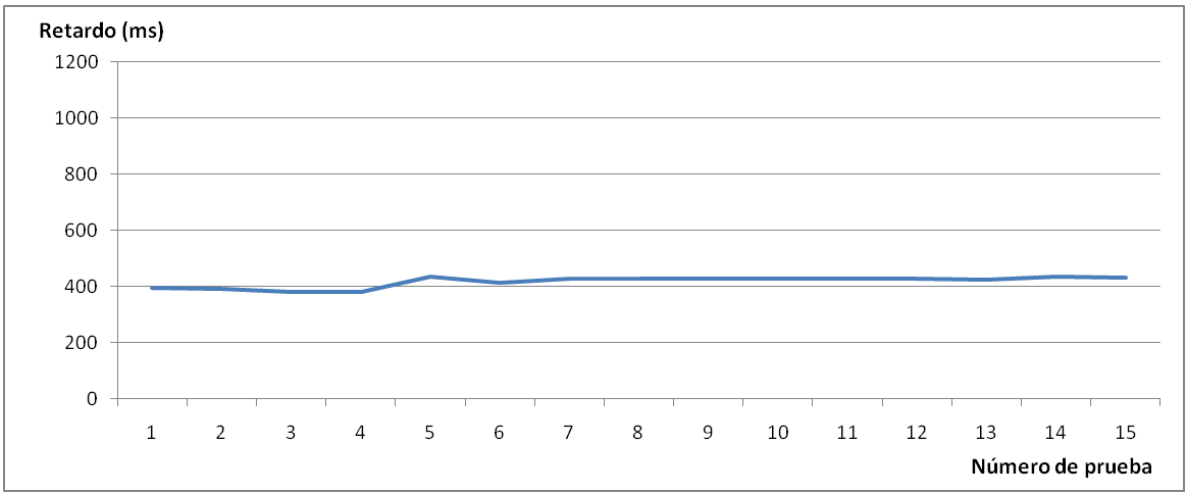

Figura 50. Retardo para audio AMR con paylaod de 250 bytes en red WiFi.

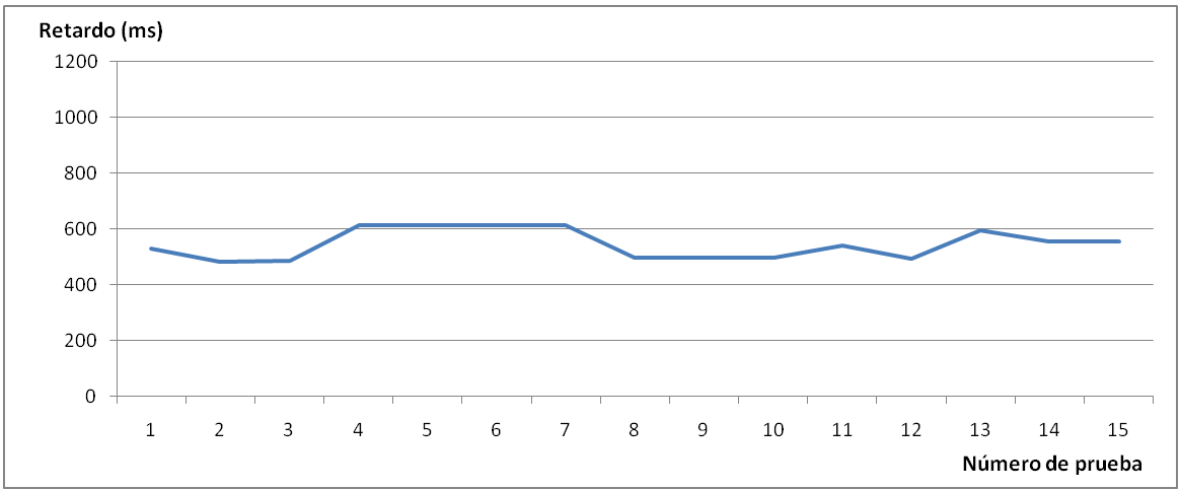

Figura 51. Retardo para audio AMR con paylaod de 500 bytes en red WiFi.

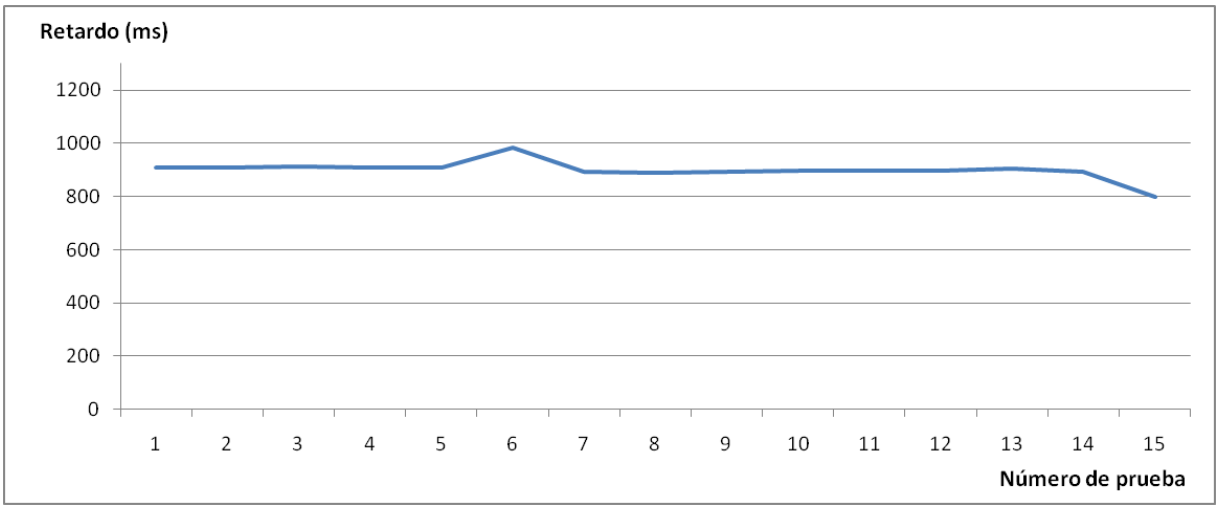

Figura 52. Retardo para audio AMR con paylaod de 1000 bytes en red WiFi.

En este caso, también se ha vuelto a experimentar que conforme aumenta el tamaño del payload para los casos de uso de audio en AMR, el retardo estimado aumenta aunque no ocurre lo mismo con su variabilidad.

7.4.4 Conclusiones

El retardo estimado en la red WiFi es siempre menor que el experimentado por la red de datos 3G, tanto en promedio como en su desviación típica, lo que verifica el concepto de jitter como problema únicamente presente en las redes móviles de datos.

Por los resultados obtenidos en ambos tipos de redes, el códec AMR con un tamaño de payload de 250 bytes parece ser la mejor solución por su bajo valor de retardo y poca variabilidad del mismo.

Por otro lado, sólo se ha superado el segundo de retardo para algunas medidas realizadas sobre la red 3G y para el caso de audio AMR y payload de 1000 bytes. Por tanto, la mayoría de los retardos experimentados se encuentran por debajo del segundo y con un promedio cercano a los 500 ms.

Este hecho permite concluir con seguridad que la generación de transcripciones para el servicio de subtitulado en tiempo real no se verá afectado por el uso de un dispositivo móvil que envía la voz a través de internet hasta el servidor.

Además, el proceso ASR necesita un cierto tiempo para transcribir el discurso del profesor debido a la complejidad computacional de la tarea, esperando que haya un breve silencio para proporcionar toda la transcripción. De esta manera, si el maestro habla durante mucho tiempo sin hacer pausas, las frases transcritas por la ASR son más largas y el retardo aumenta. En cambio, si el profesor hace pausas frecuentes, la calidad del sistema mejora considerablemente. Por esta razón, si existen retrasos en la generación de subtítulos, la razón puede que no sea un retardo de red.

7.5 Calidad en la transcripción audio-texto

En el proceso de pruebas no se han omitido aquellas relativas a la eficiencia del proceso de reconocimiento de voz, ya que aunque la elección del motor de reconocimiento DNS fue impuesto por la arquitectura inicial de APEINTA, es imprescindible verificar la correcta transcripción de la voz en función de la codificación elegida en el dispositivo móvil.

La evaluación de la subtitulación en tiempo real se realizó mediante una medida típica con el fin de evaluar si el servicio es adecuado: el WER de acuerdo con la Distancia de Levenshtein [91] entre el texto transcrito y la transcripción original (referencia) a nivel de palabra.

7.5.1 HResults

HResults es la herramienta de análisis proporcionada por HTK para evaluar el rendimiento de un sistema ASR. Se lee en un conjunto de ficheros con palabras (por lo general, son la salida de una herramienta de reconocimiento como *Hvite*) y los compara con los archivos de transcripción de referencia correspondientes; es decir, compara las etiquetas de entrada y las de salida, dando lugar a una matriz de confusión de los distintos HMM.

La comparación se basa en un procedimiento de alineamiento de cadena de texto, utilizando la métrica estándar estadounidense NIST FOM.

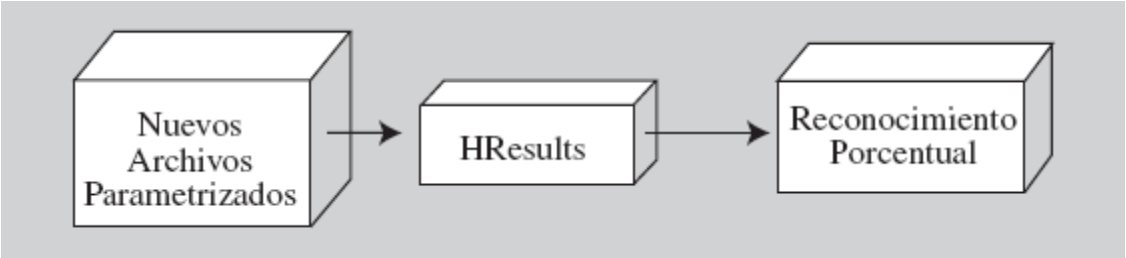

Figura 53. Diagrama de bloques de la llamada a HResults.

Se trata de una herramienta por línea de comandos, por lo que la salida se produce por pantalla y para guardar los resultados hay que usar un re direccionamiento adecuado hacia un archivo de texto.

Los resultados contienen las estadísticas de rendimiento del sistema reconocedor, como:

- Estadísticas de reconocimiento sobre un archivo base.
- Transcripciones alineadas en el tiempo.

Un ejemplo de la salida de HResults es el siguiente:

```
====================== HTK Results Analysis =======================

SENT: %Correct=0.00 [H=0, S=1, N=1]

WORD: %Corr=94.09, Acc=92.73 [H=414, D=7, S=19, I=6, N=440]

===================================================================
```

Donde:

- Inserciones (I): cuando hay una palabra transcrita que no se habla.
- Supresiones (D): cuando una palabra no está transcrita.
- Sustituciones (S): cuando una palabra se transcribe por una palabra diferente.
- Número total (N): número total de palabras transcritas.
- Número coincidencias (H): número total de palabras transcritas correctamente.

La primera línea muestra la tasa de reconocimiento de la oración, la cual no se ha calculado para estas pruebas, mientras que la segunda línea es la tasa de reconocimiento de palabras.

El número de coincidencias puede calcularse con:

$$H = N - S - D$$

El porcentaje de palabras reconocidas correctamente está dada por:

$$\%\ \text{Correcto} = \frac{H}{N}$$

Y la precisión se ha calculado con:

$$\%\ \text{Precisión} = \frac{H - I}{N}$$

7.5.2 Descripción de las pruebas

Se han empleado tres tipos de fuentes diferentes para la grabación de la voz en la realización de estas pruebas:

- Grabación de voz con un micrófono convencional conectado directamente al servidor de subtitulado.
- Grabación de voz codificada en PCM a 8 KHz con StreamDroid y envío por streaming hasta servidor de subtitulado
- Grabación de voz codificada en AMR modo 8 (12.2 Kbps) con StreamDroid y envío por streaming hasta servidor de subtitulado

Por otro lado, se realiza una transcripción de la lectura de un texto mediante DNS, con perfiles de usuario previamente entrenados para cada tipo de codificación empleada para la voz, y se almacena la salida de la transcripción.

Esta prueba se repite tres veces por cada fuente de audio, es decir, se han hecho un total de 9 lecturas del texto elegido. Para cada prueba se obtiene un fichero de salida que se compara con el de referencia usando la herramienta *HResults.*

7.5.3 Resultados

Los resultados que se han obtenido se analizan en la siguiente sección, permitiendo obtener para cada una de las 9 pruebas el porcentaje de palabras reconocidas y la precisión aplicada.

7.5.3.1 Micrófono

Prueba	H	D	S	I	N	Acierto (%)	Precisión (%)
1	414	7	19	6	440	94,09	92,73
2	409	13	18	1	440	92,95	92,73
3	413	8	19	0	440	93,86	93,86
Promedio						**93,63**	93,10

Tabla 53. Salida HResults para 3 medidas con voz grabada con micrófono

7.5.3.2 Voz codificada en PCM con StreamDroid

Prueba	H	D	S	I	N	Acierto (%)	Precisión (%)
1	401	12	27	5	440	91,14	90
2	411	7	22	10	440	93,41	91,14
3	411	10	19	3	440	93,41	92,73
Promedio						**92,65**	91,29

Tabla 54. Salida HResults para 3 medidas con voz grabada en PCM

7.5.3.3 Voz codificada en AMR con StreamDroid

Prueba	H	D	S	I	N	Acierto (%)	Precisión (%)
1	393	12	35	3	440	89,32	88,64
2	406	10	24	1	440	92,27	92,05
3	412	6	22	3	440	93,64	92,95
Promedio						**91,74**	91,21

Tabla 55. Salida HResults para 3 medidas con voz grabada en AMR

7.5.4 Conclusiones

La mayor tasa de acierto de palabras reconocidas se consigue usando el micrófono como fuente de grabación, alcanzando un 93,6%. Usando voz codificada en AMR se obtiene la peor tasa de acierto con un 91,74% mientras que utilizando audio en formato PCM se alcanza un acierto intermedio de un 92,65%.

Estos resultados eran los esperados: cuanto mayor es la calidad de la voz, mejor transcripción se consigue. Sin embargo, la diferencia es mínima y este hecho permite afirmar que usando voz codificada en AMR o PCM, la calidad de la transcripción es prácticamente la misma que la obtenida hasta ahora en el sistema APEINTA: una WER por debajo del 10% en todos los casos.

Capítulo 8. Gestión del Proyecto

Se detallan a continuación cada una de las fases de las que ha constado el proyecto, atendiendo siempre a un punto de vista de planificación o de gestión del mismo.

Se detalla la metodología, el ciclo de vida, y el plan de trabajo llevado a cabo a lo largo de la duración del mismo. También se calcula un presupuesto estimado para todo el proyecto al final del apartado.

8.1 Metodología de desarrollo

La metodología predominante que se ha decidido llevar a cabo para el desarrollo del proyecto ha sido la **metodología ágil**. Se trata de una metodología de gestión adaptativa e incremental, que permite llevar a cabo la ejecución de los proyectos de desarrollo de software adaptándose a los cambios y evolucionando de forma conjunta con el tiempo; es decir, los requisitos y soluciones evolucionan según la necesidad del proyecto. Para ello, es necesaria la colaboración de equipos muy organizados, capacitados para la toma de decisiones a corto plazo.

Se llevó a cabo un estudio previo y una fase de documentación, si entrar demasiado en detalle, para ver las necesidades del cliente y cómo resolverlas. Cada una de estas necesidades son los requisitos del sistema y se ha intentado en todo momento no modificar ni el objetivo ni el alcance dentro del ciclo de vida del proyecto de cada uno de ellos, para evitar que interfiriera con el resto. De esta forma, se han establecido diferentes etapas o fases bastante limitadas en el tiempo, de corto alcance, cada una detallada lo mínimo posible inicialmente pues en este tipo de metodología prima el funcionamiento del software frente a la documentación exhaustiva.

La comunicación con el cliente, en este caso personal del CESyA o la tutora del proyecto, se ha realizado través de email principalmente y con reuniones esporádicas.

8.2 Ciclo de vida

Se ha escogido un **ciclo de vida en cascada**, distinguiendo diferentes fases con sus tareas correspondientes en cada una. Generalmente, para cada tarea se realiza el diseño y su implementación correspondiente, y en algunos casos, un conjunto de pruebas para verificar que cumple con lo establecido.

A continuación, se analizan las principales fases planificadas:

- La primera fase corresponde a un estudio previo de sistemas similares al planteado en este proyecto mediante un *Estado del Arte* y un análisis de APEINTA.
- Durante el análisis, se verifican los requisitos funcionales y no funcionales de la aplicación, para poder adaptarlos a un diseño posterior en función la plataforma móvil elegida. También hay que tener en cuenta otras partes del sistema a desarrollar, como el tratamiento de la información en el servidor, las cuales también serán necesarias

diseñarlas más tarde. Además, se estudia un posible entorno de desarrollo y se eligen las tecnologías principales de todo el sistema.

- Una vez en fase de diseño, se propone una arquitectura completa del sistema aplicación cliente y servidor junto con el diseño de todos los requisitos previamente contemplados en la fase de análisis.
- Es el momento de la etapa de implementación, en la que se deben de llevar a la realidad todas las tareas diseñadas en la anterior fase desde el punto de vista del desarrollo. En este caso se ha implementado una parte cliente mediante una aplicación móvil Android y una parte servidor que recibe datos de la aplicación cliente.
- Por último, en la fase de pruebas, se lleva un riguroso control con una batería de pruebas unitarias, de forma que se pueda verificar que se cumplen los distintos requisitos encontrados durante el análisis.

8.3 Planificación inicial

A continuación se detalla una planificación inicial para todas las fases con las distintas tareas que se han abordado en el proyecto.

8.3.1 Tabla de fases o tareas principales

La siguiente tabla muestra las fases principales de la planificación inicial, junto con la duración de las mismas y las fechas de inicio y fin de cada una de ellas:

Nombre de fase	Comienzo	Fin	Días
Propuesta del Proyecto	01/10/10	01/10/10	1 día
Estado del arte	02/10/10	12/10/10	11 días
Estudio previo de APEINTA	13/10/10	14/10/10	2 días
Análisis	15/10/10	30/10/10	16 días
Diseño	01/11/10	15/12/10	45 días
Implementación	16/12/10	15/02/11	62 días
Fase de pruebas	02/02/11	28/02/11	27 días
Documentación	01/03/11	31/03/11	31 días

Tabla 56. Tabla de fases de Planificación inicial

8.3.2 Planificación temporal (Gantt)

Correspondiendo a la planificación inicial, se puede apreciar el reparto de tareas y su duración en días en el siguiente diagramas de Gantt (ver figura 54).

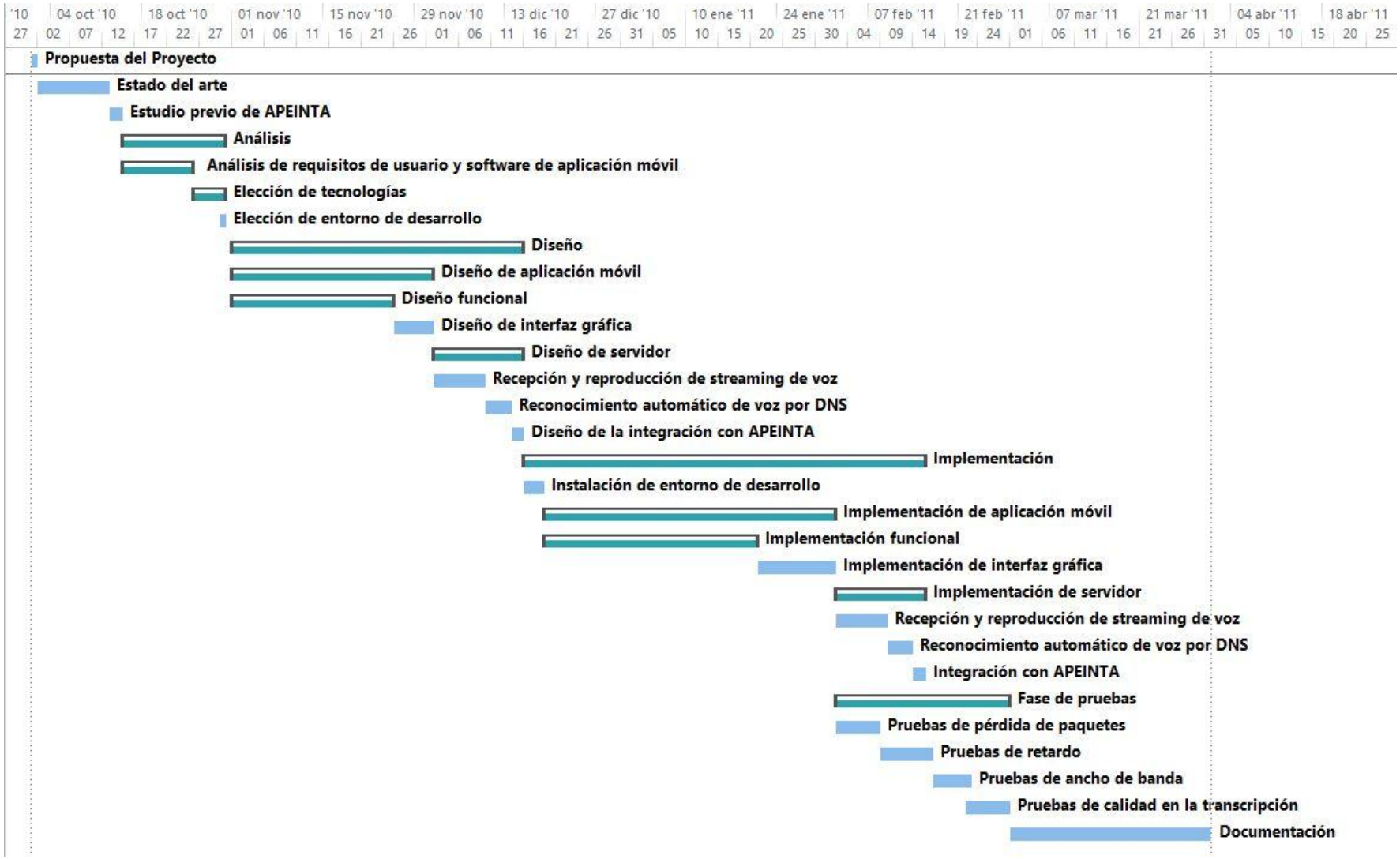

Figura 54. Diagrama de Gantt para Planificación inicial

8.3.3 Consideraciones

Una vez visto el diagrama de Gantt general con todas las fases para la planificación inicial, hay que tener en cuenta los siguientes puntos:

- El desarrollo de todas las fases del proyecto ha sido llevado a cabo por el autor, asumiendo cuando era necesario cada uno de los diferentes roles.
- Para el primer mes se planifico una media de **35 horas semanales**, es decir, 5 horas por día natural, ya que inicialmente el autor disponía de la jornada completa.
- La duración total del proyecto para esta planificación inicial es de **181 días**, unos 6 meses de duración, desde el **1 de Octubre de 2010** hasta el **31 de Marzo de 2011**.
- Hay un solapamiento de 15 días entre la última fase de Implementación y la primera parte de la fase de pruebas. La idea es que se comiencen las pruebas mientras aún se continúa en la fase de implementación.

8.4 Ejecución final y Análisis de Costes

A continuación se detalla la planificación final que realmente se han llevado a cabo para la realización del proyecto con todas las fases y sus correspondientes tareas. También se especifican los costes asociados.

8.4.1 Consideraciones iniciales

Una vez finalizado el proyecto por completo y a la vista de la planificación final resultante, se tienen las siguientes consideraciones:

- La planificación final comenzó en Octubre de 2010, al igual que la inicial, siendo de los pocos aspectos en los que ambas planificaciones coincidieron.
- Durante los primeros meses se dedicaron los días planificados a cada tarea a grandes rasgos, pero la fase de implementación se alargó mucho debido a que algunas tareas de esta fase han resultado complicadas de desarrollar.
- Destacar el retraso en la realización de la tarea "Envío de audio con compresión por streaming" tanto en la fase de diseño como en la de implementación con respecto a la planificación inicial. Para la implementación del envío de flujo de voz codificada en AMR usando RTP en la plataforma Android fue necesario usar librerías de terceros y consultar proyectos de software libre ya que no existía soporte nativo de Android en esas fechas para esta funcionalidad. Además se realizaron pruebas de viabilidad de esta característica para comprobar que se adaptaba a los objetivos definidos.
- Se esperaba finalizar el proyecto para mediados de Junio de 2011, pero por motivos laborales se interrumpió el desarrollo del proyecto el 15 de Mayo de 2011, dejando únicamente sin finalizar la documentación del mismo. Hasta ese momento, el retraso con respecto a la planificación inicial era ya de 75 días aproximadamente.
- Desde Junio de 2011, el autor ha trabajado en una empresa a jornada completa, imposibilitando la dedicación de más horas al proyecto hasta fechas recientes.

- A mediados de Agosto de 2015 se decidió retomar el proyecto implementando una serie de actualizaciones software en la aplicación Android:
  - Adaptación a nuevos formatos de pantalla y resoluciones de nuevos dispositivos.
  - Nueva interfaz gráfica para adaptarse al diseño propio de cada versión de Android.
  - Se añade soporte nativo para RTP, ya que en Abril de 2011 aún no se había lanzado la versión 3.1 de Android (donde se incluyó esta posibilidad por primera vez) y solo se podía implementar con librerías de terceros.
  - Se añade más robustez en general y mejor tolerancia a fallos.
- La actualización no ha afectado a ninguna característica del servidor pues se ha decidido continuar con el mismo que mantenía el sistema de APEINTA en 2011.
- Tras una fase de pruebas para comprobar el funcionamiento correcto de todas las nuevas actualizaciones para la aplicación, se realizó la documentación del proyecto manteniendo la misma duración que se planifico inicialmente para esta tarea.
- Los días de trabajo destinadas al proyecto a partir de Agosto de 2015 han coincidido en su mayoría con días laborales del autor.

8.4.2 Tabla de fases o tareas principales

Las fases principales de la planificación final, junto con la duración de las mismas y las fechas de inicio y fin de cada una de ellas, se identifican en la siguiente tabla:

Nombre de tarea	Comienzo	Fin	Días
Propuesta del Proyecto	01/10/10	01/10/10	1 día
Estado del arte	02/10/10	17/10/10	16 días
Estudio previo de APEINTA	18/10/10	22/10/10	5 días
Análisis	23/10/10	10/11/10	19 días
Diseño	11/11/10	31/12/10	51 días
Implementación	01/01/11	27/04/11	117 días
Fase de pruebas	15/04/11	15/05/11	31 días
Actualización	15/08/15	10/09/15	27 días
Fase de pruebas de actualización	05/09/15	16/09/15	12 días
Documentación	17/09/15	18/10/15	32 días

Tabla 57. Tabla de fases de Planificación final

8.4.3 Planificación temporal (Gantt)

También se ha diseñado un diagrama de Gantt para visualizar el reparto de tareas y duración para la planificación real en las siguientes figuras (figuras 55-56).

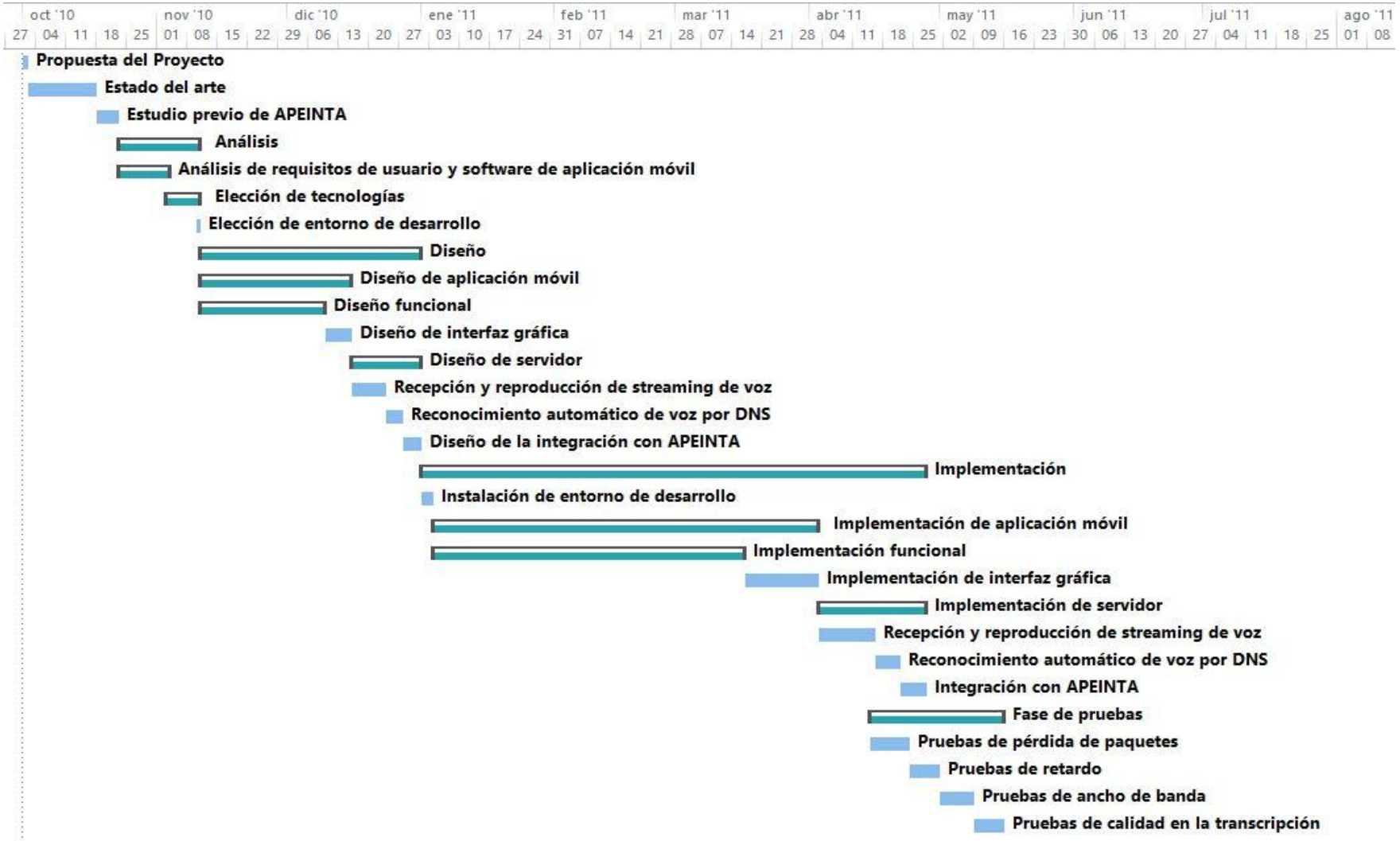

Figura 55. Diagrama de Gantt para Planificación final 2010-2011

Figura 56. Diagrama de Gantt para Planificación final 2015

Debido a la cantidad de tareas y sub-tareas de cada fase para la planificación final y teniendo en cuenta que finalmente el proyecto se ha desarrollado en dos periodos de tiempo muy distanciados, se ha decidido proporcionar un diagrama de Gantt para todas las fases y tareas de 2010-2011 y otro correspondiente sólo a la actualización de la aplicación móvil y a la preparación de la documentación para el año 2015.

8.4.4 Presupuestos

A continuación se realiza un análisis de los costes parciales y totales que ha supuesto el proyecto. A la hora de ajustar los costes, se han tenido en cuenta diversas referencias dentro del sector. Además, se han dividido los presupuestos en varias tablas.

8.4.4.1 Costes de personal

En la siguiente tabla se observan los distintos perfiles de profesionales que han intervenido en la realización de las tareas con su coste anual, mensual y por día:

Perfil	Sueldo bruto anual	Sueldo neto mensual	Sueldo neto por día
Analista/Diseñador	34000 €	1787 €	85 €
Programador	28000 €	1512 €	72 €
Evaluador	30000 €	1600 €	76 €

Tabla 58. Perfiles y costes.

Se ha asumido que la media de días laborales al mes son 21 días. La fuente de estos datos es un estudio de remuneración del 2015 realizado por *Page Personnel* [92].

En las tablas 59-64 se muestra el coste por cada perfil profesional, en total y por tarea. Además, también se indica el tiempo empleado en cada tarea:

Nombre de tarea	Tiempo	Coste
Propuesta del Proyecto	1 día	85 €
Estado del arte	16 días	1.360 €
Estudio previo de APEINTA	5 días	425 €
Análisis	19 días	1.615 €
Análisis de requisitos de usuario y software de aplicación móvil	*11 días*	*935 €*
Elección de tecnologías	*7 días*	*595 €*
Elección de entorno de desarrollo	*1 día*	*85 €*
Documentación	32 días	2.720 €
TOTAL	**73 días**	**6.205 €**

Tabla 59. Precio Analista

Nombre de tarea	Tiempo	Coste
Diseño	51 días	4.335 €
Diseño de aplicación móvil	35 días	2.975 €
Diseño funcional	*30 días*	*2.550 €*
Diseño de interfaz gráfica	*5 días*	*425 €*
Diseño de servidor	16 días	1.360 €
Recepción y reproducción de streaming de voz	*8 días*	*680 €*
Reconocimiento automático de voz por DNS	*4 días*	*340 €*
Diseño de la Integración con APEINTA	*4 días*	*340 €*
TOTAL	**51 días**	**4.335 €**

Tabla 60. Precio Diseñador

Nombre de tarea	Tiempo	Coste
Implementación	117 días	8.424 €
Instalación de entorno de desarrollo	3 días	216 €
Implementación de aplicación móvil	89 días	6.408 €
Implementación funcional	*72 días*	*5.184 €*
Implementación de la interfaz gráfica	*17 días*	*1.224 €*
Implementación de servidor	25 días	1.800 €
Recepción y reproducción de streaming de voz	*13 días*	*936 €*
Reconocimiento automático de voz por DNS	*6 días*	*432 €*
Integración con APEINTA	*6 días*	*432 €*
TOTAL	**117 días**	**8.424 €**

Tabla 61. Precio Programador (Implementación)

Nombre de tarea	Tiempo	Coste
Actualización	27 días	1.944 €
Resolución y formato de pantalla	*11 días*	*792 €*
Patrones de diseño para nuevas versiones de Android	*3 días*	*216 €*
Nuevas APIs para uso de AMR sobre RTP	*8 días*	*576 €*
Robustez y tolerancia a fallos	*5 días*	*360 €*
TOTAL	**27 días**	**1.944 €**

Tabla 62. Precio Programador (Actualización)

Nombre de tarea	Tiempo	Coste
Fase de pruebas	31 días	2.356 €
Pruebas de pérdida de paquetes	*9 días*	*684 €*
Pruebas de retardo	*7 días*	*532 €*
Pruebas de ancho de banda	*8 días*	*608 €*
Pruebas de calidad en la transcripción	*7 días*	*532 €*
TOTAL	**31 días**	**2.356 €**

Tabla 63. Precio Evaluador (Fase inicial de pruebas)

Nombre de tarea	Tiempo	Coste
Fase de pruebas de actualización	12 días	912 €
Pruebas de adaptación a pantalla en nuevos dispositivos	*6 día*	*456 €*
Validación de nuevas APIs	*4 días*	*304 €*
Pruebas genéricas para comprobación de fallos	*2 días*	*152 €*
TOTAL	**12 días**	**912 €**

Tabla 64. Precio Evaluador (Fase de pruebas de actualización)

El coste total del personal que ha intervenido en el proyecto asciende a **24.176 €**.

8.4.4.2 Costes de materiales y mantenimiento

Existen una serie de elementos hardware y software, especialmente licencias, que se han de tener en cuenta a la hora de calcular costes.

Para calcular los costes atribuidos al proyecto se puede aplicar la siguiente fórmula:

$$\text{Coste imputable} = \frac{A * C * D}{B}$$

Dónde:

- A = Precio del equipo (sin IVA)
- B = Periodo de amortización (meses)
- C = Número de meses de uso del elemento en el proyecto
- D = Porcentaje del uso dedicado al proyecto

A continuación se muestran unas tablas (tablas 65-66) con el hardware y software requerido y con el cálculo de costes imputables al proyecto asociados a cada elemento.

Material	Precio unitario	Periodo de amortización	Dedicación al proyecto	Porcentaje de uso dedicado al proyecto	Coste imputable
Nexus One	212 €	24 meses	5 meses	40%	20 €
Galaxy Nexus	240 €	24 meses	3 meses	40%	12 €
ASUS A550L	600 €	60 meses	8 meses	50%	40 €
Servidor APEINTA	739 €	60 meses	5 meses	80%	49 €
Tarjetas Audiophile	310 €	60 meses	5 meses	80%	21 €

Tabla 65. Coste imputable material utilizado (Hardware)

Material	Precio unitario	Periodo de amortización	Dedicación al proyecto	Porcentaje de uso dedicado al proyecto	Coste imputable
Microsoft Windows 8.1	119,99 €	24 meses	8 meses	50%	20 €
Microsoft Windows 7	225,99 €	24 meses	5 meses	80%	38 €
Microsoft Office 2007	685 €	60 meses	2 meses	50%	11 €
Eclipse	0 €	-	-	-	0 €
VLC	0 €	-	-	-	0 €
Android SDK	0 €	-	-	-	0 €
DNS 10.1 Professional	625 €	60 meses	5 meses	80%	42 €

Tabla 66. Coste imputable material utilizado (Software)

Por tanto, los costes de materiales y funcionamiento del proyecto ascienden a **253 €**.

8.4.4.3 Resumen de Costes

A continuación se muestra una tabla con el resumen de los presupuestos del proyecto:

Concepto	Coste total
Personal	24.176 €
Costes de materiales y mantenimiento	253 €
Costes Indirectos (20%)	4.885 €
Total (Sin IVA)	29.314 €
Total (IVA 21%)	**35.470 €**

Tabla 67. Resumen presupuestos

Se ha estimado que los costes indirectos suponen el 20% del coste restante.

8.5 Conclusiones finales

La duración final del proyecto ha variado respecto a la planificada inicialmente. La evolución de Android así como la poca disponibilidad del autor para abordar la finalización del proyecto con anterioridad, han causado que se añadiese una fase de actualización al final.

El segmento de tiempo ha sido del **1 de octubre de 2010** al **18 de octubre de 2015**, abarcando un total de **290** días de trabajo, repartidos en dos períodos de **226** días y **64** días respectivamente.

El coste total del proyecto asciende a **35.470 €,** de los de los cuales la mayor parte corresponden al personal profesional que ha realizado el proyecto. Hay que destacar el uso de aplicaciones con licencias de libre distribución, que han permitido no encarecer el presupuesto total en demasía.

Capítulo 9. Conclusiones y Trabajos Futuros

En este capítulo se repasa todo lo que se ha observado durante el desarrollo del proyecto, y especialmente se extraen conclusiones a partir de los resultados que se han obtenido. También se detallarán algunas líneas de investigación que podrían servir para continuar con este trabajo en un futuro.

9.1 Conclusiones

En el presente proyecto fin de carrera se ha estudiado la posibilidad de enviar un flujo continuo de voz registrada por el micrófono de un terminal móvil hasta el sistema de reconocimiento de voz del proyecto APEINTA, llevándose a cabo todos los objetivos marcados. Se diferencian, por tanto, dos fases del estudio:

1. **Envío de streaming de audio desde un terminal móvil a través de redes 3G y WiFi.**

Se ha realizado un estudio de los sistemas operativos móviles más relevantes y de los protocolos de transmisión de datos multimedia más típicos así como una introducción a la codificación de la voz. Por otra parte, se ha implementado una aplicación para smartphones con sistema operativo Android, denominada StreamDroid. Esta aplicación permite el envío de la señal de audio grabada por el micrófono mediante uno de estos dos tipos de streaming:

- Audio codificado en PCM a 8/16 KHz y 16 bits sobre el protocolo UDP
- Audio codificado en AMR a 8 KHz y con tasa variable sobre el protocolo RTP.

2. **Recepción de la voz por el servicio de transcripción en tiempo real de APEINTA.**

Se han estudiado inicialmente varios sistemas de subtitulado en tiempo real similares al propuesto. Posteriormente se ha diseñado e implementado un módulo para recibir el streaming de audio a través de UDP o RTP y otro para la conversión de datos. Para esto, se ha utilizado el reproductor VLC, debido a su buen soporte para protocolos de red y a su compatibilidad con un gran número de códecs, y dos tarjetas de sonido.

Finalmente, se realizaron pruebas con el sistema para comprobar su completa viabilidad tanto en redes 3G como en redes WiFi. Los resultados de estas pruebas se han analizado en función de los cuatro aspectos más relevantes:

Fiabilidad de la información entre emisor y receptor:

A pesar de usar el protocolo UDP o RTP para la transmisión, lo que no garantiza que lleguen todos los paquetes, el servidor recibe al completo el flujo de datos que se envió desde el smartphone cuando se usa la red 3G, independientemente de la calidad de su cobertura. Sin embargo, algunos paquetes de datos podrían perderse en la transmisión cuando el dispositivo se conecta a algunos tipos de redes WiFi (como la red WiFi pública probado), lo cual se traduce en pequeños cortes en la reproducción de la voz.

Por lo tanto, se puede concluir que tanto la red de datos 3G como las redes WIFI son capaces de enviar voz en streaming con gran fiabilidad en los entornos de prueba elegidos.

Ancho de banda:

Esta característica depende de la codificación de voz elegida en la aplicación móvil:

- PCM: Puede estar ligeramente por encima de 128 Kbps o 256 Kbps.
- AMR: A lo sumo, sólo supone un pequeño incremento por encima de los 12.2 Kbps.

Por los resultados obtenidos, todas las configuraciones de voz que se han probado no han presentado problemas de ancho de banda a la hora de transmitir el flujo de voz.

Retardo de red:

El retardo especifica el tiempo que emplea un paquete en ser transmitido a través de la red desde el smartphone al servidor, siendo variable en función de la calidad y estado de la red.

La mayoría de los estudios realizados en este proyecto demuestran que el retraso no es superior a 1 segundo en las redes 3G y WiFi, un valor adecuado para este tipo de servicios en tiempo real. Por otra parte, si este retraso se compara con los tiempos de procesamiento para frases largas del ASR usado por APEINTA, Dragon Naturally Speaking, resulta despreciable y la calidad del servicio no se ve afectada.

Calidad de la transcripción:

La voz codificada en PCM usa una tasa de 128 Kbps o 256 Kbps, es decir, una calidad de voz muy similar a la que consigue grabar un micrófono convencional, que era el método que hasta ahora usaba el servicio de APEINTA para registrar la voz del orador. El otro códec usado para transmitir la voz, AMR, emplea tasas del orden de 10 veces menor por lo que la calidad de la voz no resulta ser tan buena. Sin embargo, se han realizado pruebas con estos tres métodos para codificar la voz y enviarla al servidor de subtitulado de APEINTA, y el resultado es que en todos los casos la WER tras realizar una transcripción se sitúa por debajo del 10%.

Por tanto, el envío de voz desde un smartphone Android por streaming hasta el servidor de transcripción de APEINTA es totalmente viable y supone una mejora respecto a la arquitectura actual gracias a las conclusiones que se han obtenido de las pruebas y que han sido reflejadas en esta sección. De hecho, se han cumplido con todos los objetivos propuestos y además se puedan realizar nuevos desarrollos que aprovechen el trabajo realizado hasta ahora.

En otro orden de cosas, mediante la realización de este proyecto también he sido consciente de la necesidad de desarrollar tecnologías accesibles para cualquier persona con discapacidad, especialmente de aquellas que se encuentran en edad de pleno desarrollo intelectual, ya que en muchas ocasiones no se tienen en cuenta circunstancias específicas provocando limitaciones o barreras de accesibilidad. Desde mi punto de vista la inversión en tecnología sería necesaria para contemplar en mayor medida soluciones para estas posibles barreras.

Además, gracias a este proyecto he adquirido nuevos conocimientos relativos a la utilización de herramientas para la accesibilidad como la realización de transcripciones de voz a texto que permite una solución como Dragon Naturally Speaking.

9.2 Trabajos futuros

En este capítulo se expondrán algunas propuestas para el desarrollo de futuros proyectos que pueden realizarse aprovechando la investigación iniciada con éste.

Continuando con la misma línea de desarrollo de StreamDroid, hay dos posibles mejoras:

- **Compresión de audio**: la API *MediaRecorder* también soporta el uso de otros códecs como AMR-WB o varios de tipos de AAC.
- **Protocolos de red**: de forma nativa Android también permite usar RTSP y HTTP/HTTPS.

Sin embargo, existen otras líneas de investigación que se desmarcan un poco más del proyecto y que son muy interesantes, relacionadas con nuevas arquitecturas y en qué parte de las mismas se lleva a cabo el proceso ASR:

Reconocimiento de voz integrado

Se trata de integrar el módulo de grabación y codificación de la voz junto con el módulo de ASR en el mismo dispositivo móvil. Desde el punto de vista de la arquitectura de APIENTA, se integraría la función del servidor de transcripción en tiempo real dentro del smartphone.

De esta forma, el servicio no depende de una comunicación con un servidor, por lo que no existe el concepto de latencia de red, y es sustancialmente más económico pues no se requiere el servidor de subtitulado de APEINTA. En contrapartida, no se pueden realizar cálculos complejos por limitaciones de procesamiento y memoria.

Reconocimiento de voz en la nube

Esta solución también permitiría prescindir del servidor actual de APEINTA, pues el reconocimiento de voz se llevaría a cabo en la nube y el smartphone sería el encargado de distribuir los subtítulos recibidos.

Podría mejorar la velocidad de generación y precisión de las transcripciones en función del servicio contratado, y no sería necesario actualizar ni mantener el software ASR. Una de las desventajas podría ser el mayor uso de datos móviles en el smartphone, pues ahora no sólo se enviaría la voz sino que también necesitaría recibir los resultados de las transcripciones.

A continuación se listan algunas soluciones reales para implementar estas nuevas propuestas:

- Uso de **APIs nativas** disponibles en el SDK de Android para el reconocimiento de voz.
- Uso de las librerías de ***Pocketsphinx***, un proyecto derivado de CMU Sphinx.
- ***MyCaption*** [93] proporciona un servicio HTTP para enviar audio y recibir transcripciones en texto, cuyo precio varía en función del tiempo de audio procesado.
- Uso de la nube de ***iSpeech*** [94] para un servicio de ASR mediante un SDK para Android específico, el cual soporta 27 idiomas.
- ***AT&T*** [95] tiene disponible un SDK para Android 2.3 o superior que permite realizar procesos de ASR en la nube mediante un conjunto de bibliotecas Java.
- ***Dragon*** SDK para Android proporciona una integración sencilla de servicios de reconocimiento de voz en esta plataforma.

Capítulo 10. Bibliografía.

[1] *Centro español de Subtitulado y audiodescripción.* http://www.cesya.es.

[2] CESyA, Universidad Carlos III de Madrid. «Proyecto APEINTA, Apuesta por la Enseñanza Inclusiva. Uso de Nuevas Tecnologias dentro y fuera del aula.» 2009.

[3] *Web oficial de la UC3M.* http://www.uc3m.es.

[4] Javier Jiménez, Ana Iglesias, Juan Francisco López, Belén Ruiz-Mezcua, Julián Hernández. «Tablet PC and Head Mounted Display for Live Closed Captioning in Education.» 29th IEEE International Conference on Consumer Electronics (ICCE). Las Vegas, U.S.A, January, 2011.

[5] Mark Kalma, Eckehard Steinbach, Bernd Girod. «Adaptive media playout for low-delay video streaming over error-prone channels.» En *Circuits and Systems for Video Technology*. Dept. of Electr. Eng., Stanford Univ., CA, USA, 2004.

[6] *Web oficial de IDC.* http://www.idc.com/.

[7] *Web oficial de Android.* http://www.android.com/.

[8] *Web oficial de Google.* http://www.google.es/intl/es/about.html.

[9] *Web oficial de OHA.* http://www.openhandsetalliance.com/.

[10] *Web oficial de Google Play Store.* https://play.google.com/store.

[11] *Web oficial de QUEMU.* http://www.qemu.org/.

[12] *Web oficial de Eclipse.* http://www.eclipse.org/.

[13] *Web oficial de ADT.* http://developer.android.com/tools/sdk/eclipse-adt.html.

[14] *Web oficial de Android Studio.* http://developer.android.com/tools/studio/index.html.

[15] *Web oficial de Apache Cordova.* https://cordova.apache.org/.

[16] *Web oficial de NDK Android.* http://developer.android.com/tools/sdk/ndk/index.html.

[17] *Apache License, Version 2.0.* http://www.apache.org/licenses/LICENSE-2.0.

[18] *Web oficial de Blackberry.* http://global.blackberry.com/es.html.

[19] *Web oficial de BlackBerry App World.* http://appworld.blackberry.com/webstore.

[20] *Web oficial de Apple.* http://www.apple.com.

[21] *Web oficial de Xcode.* http://developer.apple.com/xcode/.

[22] *Web oficial de Windows Phone.* http://www.windowsphone.com/es-es.

[23] *Web oficial de Microsoft.* http://www.microsoft.com.

[24] *Web oficial de desarrollo de Windows 10.* https://dev.windows.com/es-es.

[25] *Tienda oficial de Windows Phone.* http://www.windowsphone.com/es-pe/store.

[26] «Encuesta de Discapacidad, Autonomía personal y situaciones de Dependencia (EDAD).» INE, 2008.

[27] Castro, Mercedes de. «Introducción al subtitulado en tiempo real.» Marzo 2011.

[28] López, J. Francisco. «Herramientas propietarias y gratuitas de subtitulado.» En *Máster en Tecnologías de Apoyo, Accesibilidad y Diseño para todos*. Leganés (Madrid), 2011.

[29] Cole, R. A. «Survey of the State of the Art in Human Language Technology.» Cambridge University Press, 1997.

[30] Rabiner, L. *An introduction to hidden Markov models.* AT&T Bell Laboratories. Murray Hill, New Jersey , 1986.

[31] *Web oficial de DNS.* http://www.nuance.es/dragon/index.htm.

[32] *Web oficial de Nuance.* http://www.nuance.es/.

[33] *Web Dragon para MAC.* http://www.nuance.com/for-individuals/by-product/dragon-for-mac/index.htm.

[34] *Tienda oficial Nuance.* http://shop.nuance.es/store/nuanceeu/es_ES/DisplayHomePage.

[35] *Web oficial ViaVoice.* http://www-01.ibm.com/software/pervasive/viavoice.html.

[36] *Web oficial de IBM.* http://www.ibm.com/es/es/.

[37] *Web oficial Media Mining Indexer.* http://www.sail-labs.com/products-solutions/commercial-products/media-mining-indexer.html.

[38] *Web oficial de Sail Labs.* http://www.sail-labs.com/home.html.

[39] *Web oficial de DynaSpeak.* http://www.speechatsri.com/products/dynaspeak.shtml.

[40] *Web oficial de SRI International.* http://www.sri.com/.

[41] *Web oficial de LumenVox.* http://www.lumenvox.com/.

[42] *Web oficial de Verbio.* http://www.verbio.com/.

[43] *Web oficial de CMU Sphinx.* http://cmusphinx.sourceforge.net/.

[44] *Web oficial de Julius.* http://julius.osdn.jp/en_index.php.

[45] *Web oficial de Voxforge.* http://www.voxforge.org/es.

[46] *Web oficial de HTK.* http://htk.eng.cam.ac.uk/.

[47] *Web oficial en español de Google Search.* http://www.google.com/landing/now/.

[48] *Web oficial de Siri.* http://www.apple.com/es/ios/siri.

[49] *Web oficial de Cortana para Windows Phone.* http://www.windowsphone.com/en-us/how-to/wp8/cortana/meet-cortana.

[50] *Web oficial de Hound.* http://www.soundhound.com/hound.

[51] *Web de Samsung para S Voice.* http://www.samsung.com/global/galaxys3/svoice.html.

[52] *Web oficial de Sirius.* http://sirius.clarity-lab.org/.

[53] *Web oficial de Productos Dragon para móviles.* http://www.dragonmobileapps.com/.

[54] *Web oficial de Ai-Live.* https://uk.ai-live.com/.

[55] *Web oficial de Hamilton CapTel.* http://www.hamiltoncaptel.com/.

[56] Bumbalek, Zdenek, Jan Zelenka, y Lukas Kencl. «E-Scribe: Ubiquitous Real-Time Speech Transcription for the Hearing-Impaired.» Telecommunications Engineering, Faculty of Electrical Engineering, Czech Technical University, Praga.

[57] *Web oficial de Liberated Learning.* http://liberatedlearning.com/.

[58] *Liberated Learning Youth Initiative.* http://www.transcribeyourclass.ca/.

[59] Vertanen, Keith, y Per Ola Kristensson. «Parakeet: A Continuous Speech Recognition System for Mobile Touch-Screen Devices.» University of Cambridge, Cambridge, UK.

[60] Matthews, Tara, Scott Carter, Carol Pai, Janette Fong, y Jennifer Mankoff. «Scribe4Me: Evaluating a Mobile Sound Transcription Tool for the Deaf.» University of California, Berkeley, California.

[61] *Web oficial de AENOR.* http://www.aenor.es/.

[62] *Web oficial de SipDroid.* http://sipdroid.org/.

[63] Iglesias, Ana, Javier Jiménez, Pablo Revuelta, y Lourdes Moreno. «Avoiding communication barriers in the classroom: the APEINTA project.» Computer Science Department, Carlos III of Madrid University, 2014.

[64] *Web oficial de W3C.* http://www.w3c.es/.

[65] Portnoff, Michael R. *Implementation of the digital phase vocoder using the fast Fourier transform.* Massachusetts Institute of Technology, Cambridge, MA, 1976.

[66] *Web oficial de la UIT.* http://www.itu.int/es.

[67] Jayant, N. *Digital coding of speech waveforms: PCM, DPCM, and DM quantizers.* Bell Laboratories, Murray Hill, N. J., 1974.

[68] Chu, Wai C. *Speech Coding Algorithms: Foundation and Evolution of Standardized Coders.* DoCoMo USA Labs. San Jose, California, 2003.

[69] *Web oficial de 3GPP.* http://www.3gpp.org/.

[70] *Web oficial de ETSI.* http://www.etsi.org/.

[71] *Especificaciones PacketCable.* http://cablelabs.com/specifications-library/packetcable/.

[72] Holma, H, J. Melero, J. Vainio, T. Halonen, y J. Makinen. «Performance of adaptive multirate (AMR) voice in GSM and WCDMA.» Nokia Networks, Finlandia, 2003.

[73] «AMR Speech Codec Frame Structure (3GPP TS 26.101).» 3GPP, Valbonne. Francia.

[74] Postel, J. «RFC 768.» IETF, August 1980.

[75] Schulzrinne, H., S. Casner, R. Frederick, y V. Jacobson. «RFC 3550.» IETF, July 2003.

[76] Schulzrinne, H. «RFC 1890.» IETF, January 1996.

[77] Schulzrinne, H., A. Rao, y R. Lanphier. «RFC 2326.» IETF, April 1998.

[78] *Web oficial de FFmpeg.* http://ffmpeg.org/.

[79] *Licencia y legalidad FFmpeg.* http://www.ffmpeg.org/legal.html.

[80] *Lista proyectos con FFmpeg.* https://trac.ffmpeg.org/wiki/Projects.

[81] *Web oficial de Quicktime.* http://www.apple.com/es/quicktime/what-is/.

[82] *Web oficial de RealPlayer.* http://es.real.com/.

[83] *Web oficial de RealNetworks.* http://www.realnetworks.com/.

[84] *Web oficial de VLC.* http://www.videolan.org/vlc/.

[85] *Web oficial de VideoLAN.* http://www.videolan.org/.

[86] *Línea de comandos VLC.* https://wiki.videolan.org/VLC_command-line_help.

[87] Sjoberg, J., M. Westerlund, A. Lakaniemi, y Q. Xie. «RFC 4867.» IETF, April 2007.

[88] *Web oficial de GStreamer.* http://www.gstreamer.com/.

[89] *Web oficial de PacketVideo.* http://www.pv.com/.

[90] *Web oficial de Wireshark.* https://www.wireshark.org/.

[91] Levenshtein, V. L. «Binary codes capable of correcting deletions, insertions, and reversals.» 707–710. Soviet Physics Doklad, 1966.

[92] *Web oficial Page Personnel.* http://www.pagepersonnel.es.

[93] *Web oficial de MyCaption.* http://mycaption.com.

[94] *Web oficial de iSpeech.* https://www.ispeech.org.

[95] *Web de desarrolladores de AT&T.* http://developer.att.com/.

Capítulo 11. Anexos.

11.1 Glosario de siglas y acrónimos.

- **3GP**: Formato multimedia propietario de 3GPP.
- **3GPP**: 3rd Generation Partnership Project.
- **A/D**: Analógico-Digital.
- **AAC**: Advanced Audio Coding (Codificador avanzado de audio).
- **AAC-LC**: Advanced Audio Coding Low-Complexity (Codificador avanzado de audio de baja complejidad).
- **ABR**: Average bit rate (Tasa de bits media).
- **ACELP**: Algebraic Code-Excited Linear Prediction (predicción lineal de código algebraico).
- **ADPCM**: Adaptive Differential Pulse Code Modulation (Modulación por impulsos codificados diferencial y adaptativa).
- **ADT**: Android Development Tools (Herramientas de desarrollo de Android).
- **AJAX**: Asynchronous JavaScript And XML (JavaScript asíncrono y XML).
- **AMR**: Adaptive Multi-Rate (Multi-tasa adaptativo).
- **AMR-NB**: Adaptive Multi-Rate Narrowband (Multi-tasa adaptativo en banda estrecha).
- **AMR-WB**: Adaptive Multi-Rate Wideband (Multi-tasa adaptativo en banda ancha).
- **APEINTA**: Apuesta por la Enseñanza Inclusiva dentro y fuera del Aula.
- **API**: Application Programming Interface (Interfaz de programación de aplicaciones).
- **ARM**: Advanced RISC Machines (Arquitectura RISC de 32 bits).
- **ART**: Android Runtime (Tiempo de ejecución de Android).
- **ASR**: Automatic Speech Recognition (Reconocimiento automático del habla).
- **AVC**: Advanced Video Coding (Códec de Vídeo Avanzado).
- **AVD**: Android Virtual Device (Dispositivo virtual de Android).
- **BSD**: Berkeley Software Distribution (Distribución de software berkeley).
- **C++:** lenguaje de programación orientado a obejtos que deriva de C.
- **C#**: C Shar (lenguaje de programación que deriva de C/C++ y. NET).
- **CBR**: Constant bit rate (Tasa de bits constante).
- **CCITT**: Comité Consultivo Internacional Telegráfico y Telefónico.
- **CESyA**: Centro Español de Subtitulado y Audiodescripción.
- **CMR**: Codec Mode Request (Modo de codificación).
- **CNG**: Confort Noise Generation (generación de ruido de confort).
- **CS-ACELP**: Conjugate Structure Algebraic Code-Excited Linear Prediction (predicción lineal de código algebraico excitado en estructura conjugada).
- **CSR**: Contributor Source (Identificador de fuente contribuyente)
- **CSRC**: Contributor Source Counter (contador de fuente contribuyente).
- **CTN**: Comité Técnico de Normalización.
- **CUED**: Cambridge University Engineering Department (Departamento de Ingeniería de la Universidad de Cambridge).

- **DARPA**: Defense Advanced Research Projects Agency (Agencia de Proyectos de Investigación Avanzados de Defensa).
- **Dex**: Dalvik Executable (Formato ejecutable para Dalvik).
- **DNS**: Dragon Naturally System
- **DPCM**: Differential pulse-code modulation (Modulación por impulsos codificados diferencial).
- **DS0**: Digital Signal 0 (Señal digital 0).
- **DTX**: Discontinuous transmission (Transmisión discontinua).
- **DVB**: Digital Video Broadcasting (Difusión de Video Digital).
- **DVD**: Digital Versatile Disc (Disco Versátil Digital).
- **ECJ**: Evolutionary Computation Java (Computación Evolutiva de Java).
- **EDGE**: Enhanced Data Rates for GSM Evolution (Tasas de datos mejoradas para la evolución de GSM).
- **EFR**: Enhanced Full Rate (Mejora de Tasa máxima).
- **ETSI**: European Telecommunications Standards Institute (Instituto Europeo de Normas de Telecomunicación).
- **FR**: Full rate (Tasa máxima).
- **FTP**: File Transfer Protocol (Protocolo de Transferencia de Archivos).
- **GNU**: GNU is Not Unix (GNU No es Unix).
- **GPL**: General Public License (Licencia general pública).
- **GPRS**: General Packet Radio Service (Servicio general de paquetes vía radio).
- **GSM**: Global System for Mobile Communications (Sistema global para las comunicaciones móviles).
- **HMM**: Hidden Markov Model (modelo oculto de Márkov).
- **HR**: Half rate (Tasa no máxima).
- **HTML**: HyperText Markup Language (Lenguaje de Marcado de Hipertexto).
- **HTK**: Hidden Markov Model Toolkit (Modelo Oculto de Markov Toolkit).
- **HTS**: Hosted Transcription System (Sistema de Transcripción Alojado).
- **HTTP**: Hiper-text Transfer protocol (Protocolo de transferencia de Hiper-Texto).
- **IA-32**: Intel Architecture 32-bit (Arquitectura Intel de 32 bits).
- **IBM**: International Business Machines Corporation.
- **IDC**: International Data Corporation (proveedor global de inteligencia de mercado).
- **IDE**: Integrated development environmet (Entorno de desarrollo integrado).
- **IETF**: Internet Engineering Task Force (Grupo de Trabajo de Ingeniería de Internet).
- **INE**: Instituto Nacional de Estadística.
- **iOS**: iPhone OS (Sistema operativo móvil para dispositivos Apple).
- **IP**: Internet Protocol (Protocolo de Internet).
- **JDT**: Java Development Toolkit (Conjunto de herramientas de desarrollo de Java).
- **JNI**: Java Native Interface (Interfaz nativa de Java).
- **LAN**: Local Area Network (Red de área local).
- **LGPL**: Lesser General Public License (Licencia general pública menor).
- **LVCSR**: Large Vocabulary Continuous Speech Recognition (Reconocimiento de habla continua de gran vocabulario).
- **MAC OS**: Macintosh Operating System (Sistema Operativo de Macintosh).
- **MMI**: Media Mining Indexer

- **MP3**: MPEG-1 Audio Layer III (Formato de audio del estándar MPEG-1).
- **MP4**: MPEG-4 parte 14.
- **MOS**: Mean opinion score (Factor calidad de voz).
- **MPEG**: Moving Pictures Experts Group (Grupo de expertos para establecer estándares de audio y video).
- **MRCP**: Media Resource Control Protocol (Protocolo de control de recursos de media).
- **NAT**: Network Address Translation (Traducción de direcciones de red).
- **NDK**: Native Development Kit (kit de desarrollo nativo).
- **NLU**: Natural language understanding (Entendimiento de Lenguaje Natural).
- **OHA**: Open Handset Alliance.
- **OS**: Operating System (Sistema operativo).
- **PCM**: Pulse Code modulation (modulación por impulsos codificados).
- **POF**: Probabilistic Optimum Filter (Filtrado óptimo probabilístico).
- **QEMU**: Quick Emulator (Emulador rápido).
- **QoS**: Quality os Service (Calidad del Servicio).
- **QTFF**: QuickTime File Format (Fichero formato de Quicktime).
- **RDSI**: Red Digital de Servicios Integrados.
- **RF_AC**: Requisitos funcionales de acceso a la aplicación
- **RF_CF**: Requisitos funcionales de configuración de la aplicación.
- **RF_CG**: Requisitos funcionales de control de la grabación.
- **RF_PI**: Requisitos funcionales de presentación de información por pantalla.
- **RFA**: Requisito funcional de Aplicación.
- **RIM**: Research In Motion.
- **RISC**: Reduced Instruction Set Computer (Ordenador con Conjunto Reducido de Instrucciones).
- **RNFR**: Requisito no funcional de rendimiento
- **RNFU**: Requisito no funcional de usabilidad
- **RTCP**: RTP Control Protocol (Protocolo de control de RTP).
- **RTP**: Real-time Transport Protocol (Protocolo de Transporte de Tiempo real).
- **RTSP**: Real Time Streaming Protocol (Protocolo de flujo de datos en tiempo real).
- **SAIL LABS**: Speech, Artificial Intelligence and Language Labs.
- **SDK**: Software Development Kit (Herramientas de desarrollo de software).
- **SDL**: Simple DirectMedia Layer (Bibliotecas simples de DirectMedia).
- **SETSI**: Secretaría de Estado de Telecomunicaciones y para la Sociedad de la Información.
- **SID**: Silence Insertion Descriptor (Descriptor de inserción de silencios).
- **SIP**: Session Initiation Protocol (Protocolo de Inicio de Sesión).
- **SLM**: Statistical Language Modeling (Modelos de Leguaje Estadísticos).
- **SMS**: Short Message Service (Servicio de mensajeria corta).
- **S/PDIF**: Sony/Philips Digital Interface Format (Formato de Interfaz Digital Sony/Philips).
- **SSRC**: Synchronization source identifier (identificador de fuente de sincronización).
- **STFT**: Short-time Fourier transform (Transformada de Fourier de Tiempo Corto).
- **STUN**: Session Traversal Utilities for NAT (Utilidades para sesion transversal de NAT).
- **TCP**: Transmission Control Protocol (Protocolo de Control de Transmisión).
- **TLS**: Transport Layer Security (seguridad de la capa de transporte).

- **TTS**: Text To Speech (Conversión de texto a voz).
- **UDP**: User Datagram Protocol (Protocolo de datagramas de usuario).
- **UIT**: Unión Internacional de Telecomunicaciones.
- **URI**: Uniform Resource Identifierid (Identificador de recursos uniforme).
- **URL**: Uniform Resource Locator (Localizador Uniforme de recursos).
- **VAD**: Voice activity detection (Detección de actividad vocal).
- **VBR**: Variable bit rate (Tasa de bits variable).
- **VLSI**: Very Large Scale Integration (Integración a escala muy grande).
- **VoIP**: Voice over IP (Voz sobre IP).
- **W3C**: World Wide Web Consortium.
- **WAN**: Wide area network (Red de área amplia).
- **WAV**: Waveform Audio File Format (Formato de audio Waveform)
- **WCDMA**: Wideband Code Division Multiple Access (Acceso múltiple por división de código de banda ancha).
- **WER**: Word Error Rates (Tasa de error de palabra).
- **XAML**: eXtensible Application Markup Language (Lenguaje Extensible de Formato para Aplicaciones).
- **XML**: eXtensible Markup Language (Lenguaje de marcas extensible).

11.2 Guía de Usuario.

Este anexo constituye una guía de uso y configuración del sistema presentado para el proyecto fin de carrera de Álvaro Martín titulado "Reconocimiento automático de voz en APEINTA desde terminales móviles".

Se facilita al usuario un DVD con todo el software e instrucciones necesarias.

11.2.1 Instalación

11.2.1.1 Instalación de la aplicación en dispositivo móvil: StreamDroid

Se trata de una aplicación para dispositivos con sistema operativo Android que se ha desarrollado con el único propósito de enviar la voz mediante la tecnología VoIP a través de Internet constituyendo así una parte imprescindible del sistema descrito en la introducción para el proyecto fin de carrera.

11.2.1.1.1 Requisitos

De cara a poder instalar esta aplicación en el terminal móvil y así poder demostrar el objetivo principal de este estudio (envío de audio por streaming desde el terminal a un servidor remoto), son imprescindibles los siguientes requisitos:

Software

Es necesario disponer de un terminal móvil con el Sistema Operativo Android.

La versión mínima para que la aplicación pueda funcionar es Android 2.2.x, correspondiente al nivel 8 del API. Además es válida para cualquier marca o modelo de terminal.

Hardware

Hay tres requisitos indispensables hardware para ejecutar *StreamDroid* en un terminal:

- Micrófono interno para la grabación de la voz.
- Soporte operativo para conexión a Internet mediante redes móviles o WiFi.
- Soporte para Interfaz táctil.

11.2.1.1.2 Instalación

En el DVD se proporciona un fichero con extensión APK, una variante del formato JAR de Java para distribuir e instalar componentes empaquetados para la plataforma Android.

Previamente a instalar el fichero APK, es necesario activar una opción en el smartphone que permita la instalación de aplicaciones de terceros dado que no se trata de una aplicación disponible en el Google Play (tienda oficial de aplicaciones de Android).

Para ello, el usuario debe acceder a Ajustes → Seguridad → Administración de dispositivos → Orígenes (fuentes) desconocidos y habilitar la opción tal y como se indica en la figura 57.

Una vez realizado este paso, el usuario debe almacenar dentro del sistema de ficheros del smartphone el fichero de instalación APK del DVD ("StreamDroid.apk"). Para traspasar ficheros del DVD al smartphone el usuario puede usar un cable USB que conecte el smartphone al PC.

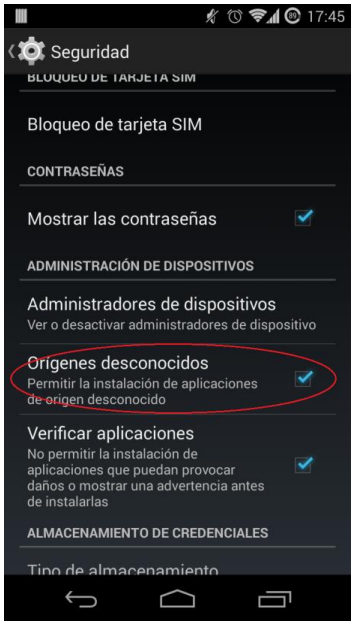

Figura 57. Activar Orígenes desconocidos en Android.

Ahora debe seleccionarse el fichero desde el explorador de archivos del smartphone, buscando en el directorio apropiado. Tras esta acción, se deberá confirmar la instalación (ver figura 58) y esperar unos segundos hasta que el sistema indique que la aplicación se encuentra instalada. Posteriormente, se deberá pulsar la opción "Listo" (ver figura 58) para terminar finalmente con el proceso de instalación o "Abrir" para iniciar la aplicación.

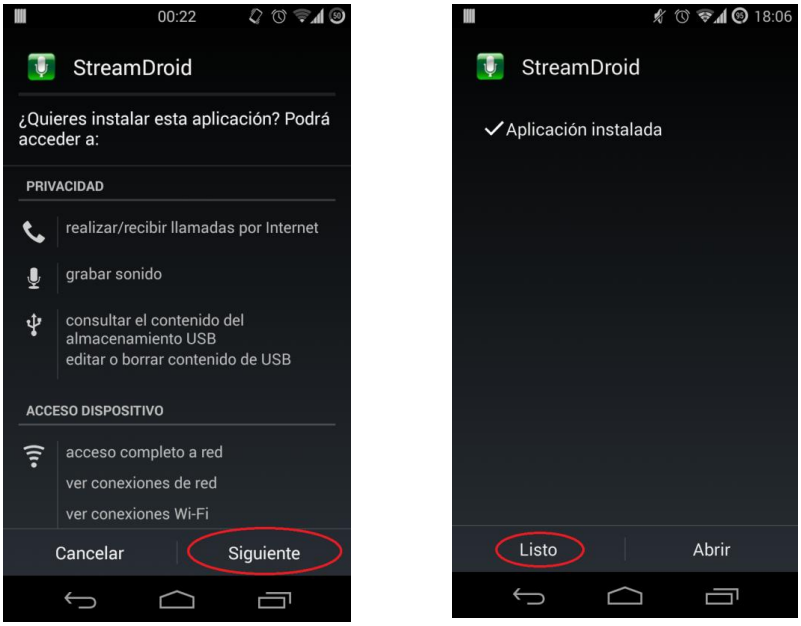

Figura 58. Proceso de instalación de StreamDroid.

11.2.1.2 Instalación del servidor de reconocimiento de voz en PC: Dragon Naturally Speaking

El software para Reconocimiento Automático de Habla que se está utilizando en el servidor de subtitulado es DNS y la versión que se ha utilizado para el desarrollo de proyecto es la versión 10.1 Profesional, que presenta algunas limitaciones.

La limitación principal de la versión elegida es el hecho de que no soporta de forma directa recibir como entrada a su sistema audio en streaming. Otras soluciones de DNS como la versión server no presentan este problema y son capaces de procesar directamente el flujo de audio desde Internet como entrada, pero debido a su precio se ha desestimado finalmente. Sin embargo, existen alternativas para conseguir esta funcionalidad de la versión server en la versión cliente. Una de ellas se explicará en la sección de configuración del servidor.

11.2.1.2.1 Requisitos

En cualquier caso, los requisitos a nivel de hardware y software para instalar y ejecutar este software de pago en el servidor de subtitulado son los siguientes:

Software

Las siguientes versiones de Microsoft Windows son compatibles:

- Windows Server 2000
- Windows Server 2003
- Windows XP (SP2 and SP3, 32-bit)
- Windows 2000 SP4
- Windows Vista™ (SP1 y SP2, 32-bit, 64-bit)
- Windows 7 (32-bit y 64-bit)
- Windows 8 (32-bit y 64-bit)

Hardware

Especificaciones mínimas del Servidor**:**

- CPU: Intel® Pentium® 4 o posterior, o AMD Athlon 64 1 GHz o posterior.
- 512 MB de RAM (1 GB de RAM para Windows Vista™ y 2 GB de RAM para Windows 7)
- Espacio libre en disco duro: 2 GB
- Caché L2: 512 KB

Especificaciones recomendadas del Servidor**:**

- CPU: Procesador Intel® Pentium® 4 / 2.4 GHz (de doble núcleo de 1.6 GHz) o procesador AMD equivalente.
- 1 GB de RAM (2 GB RAM en Windows 7 32 bits y 4 GB de RAM en Windows 7 64 bits)
- Caché L2: 1 MB

11.2.1.2.2 Instalación

Se debe ejecutar el fichero "DNSv10.1.exe" del DVD de instalación en el servidor de subtitulado siguiendo cada uno de los pasos tal y como se explica a continuación:

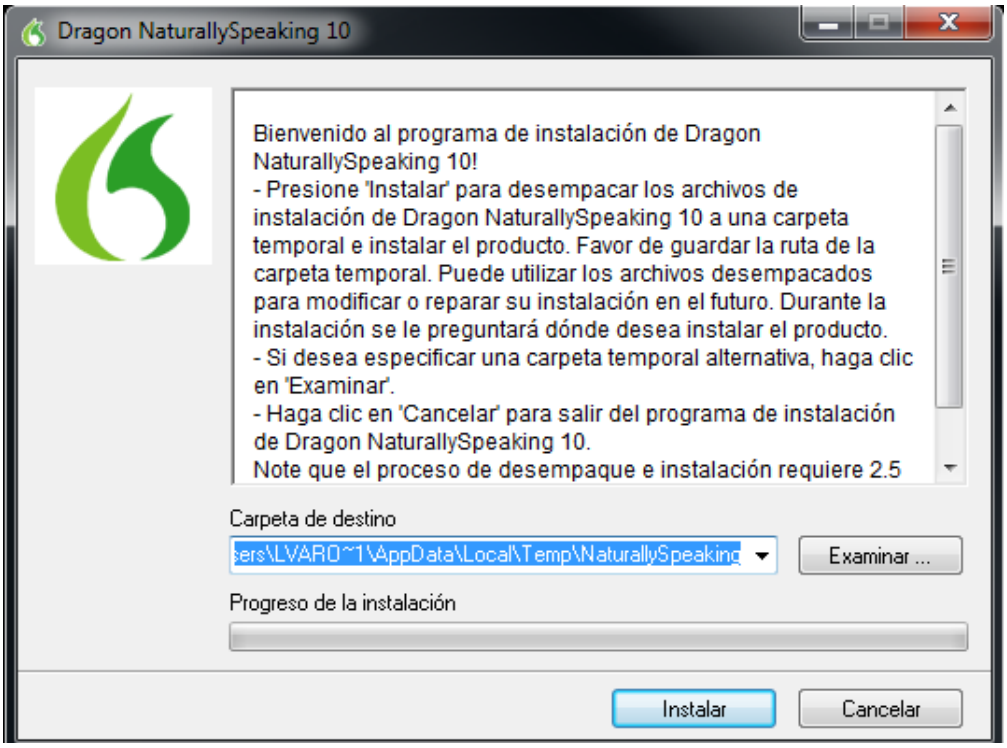

Figura 59. Proceso de Instalación de DNS.

Figura 60. Proceso de Instalación de DNS.

Se aceptan los términos de contrato de licencia y se pulsa “Siguiente” (ver figura 61).

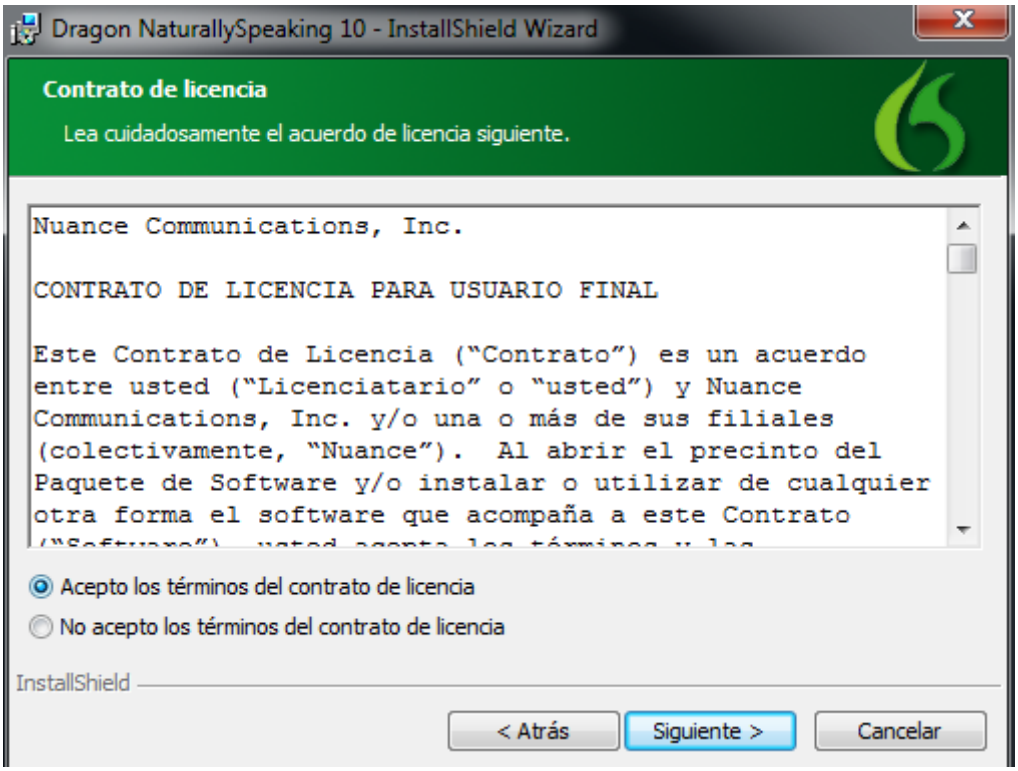

Figura 61. Proceso de Instalación de DNS.

Es importante introducir un número de serie válido (ver figura 62), como los que se proporcionan en el fichero de texto “DNSv10.1 Serials.txt” del DVD.

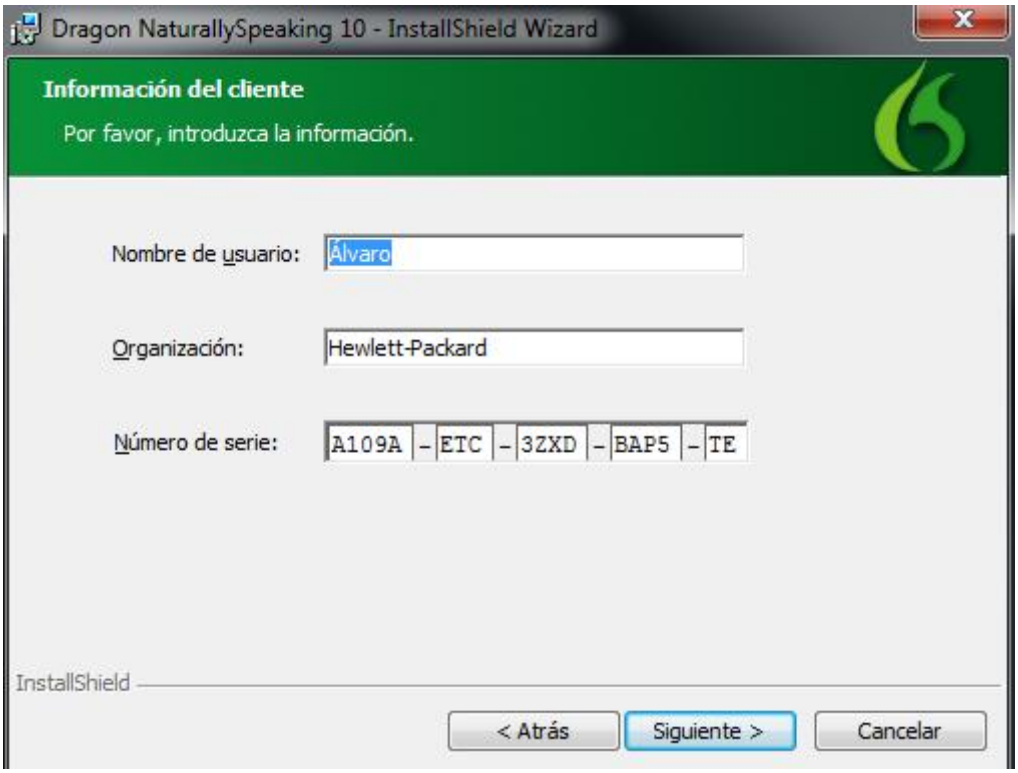

Figura 62. Proceso de Instalación de DNS.

Se debe elegir el tipo de instalación "Típica/Completa", pues la personalizada no ofrece ninguna variante que sea de utilidad para este proyecto (ver figura 63):

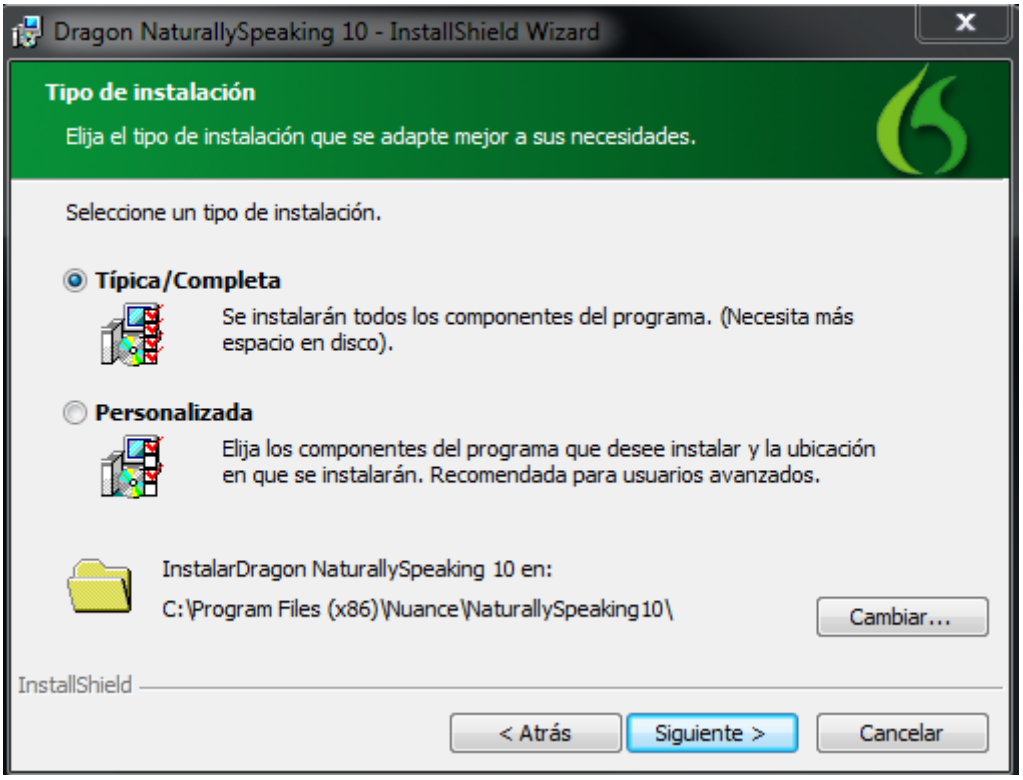

Figura 63. Proceso de Instalación de DNS.

Se activa el modo de "Ejecución Rápida", en caso de que no estuviera ya activo (ver figura 64):

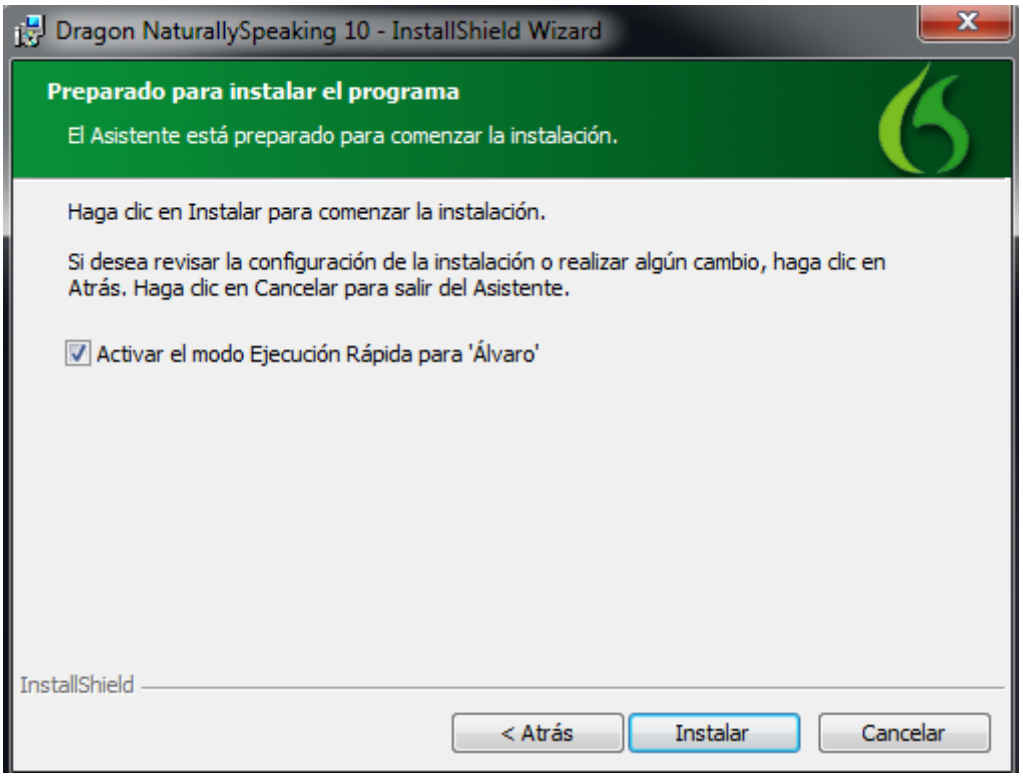

Figura 64. Proceso de Instalación de DNS.

Se espera a que finalice el proceso de instalación (ver figura 65).

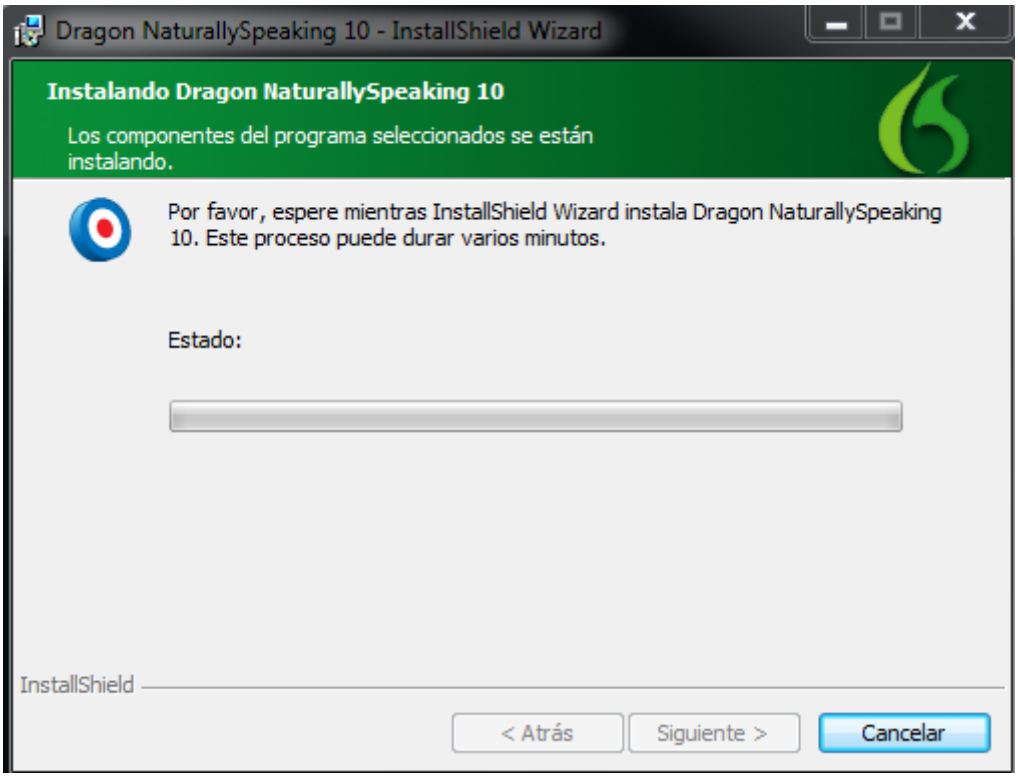

Figura 65. Proceso de Instalación de DNS.

Una vez finalizado el proceso, elegir la opción "Finalizar" (ver figura 66).

Figura 66. Proceso de Instalación de DNS.

11.2.1.3 Instalación de receptor de audio: VLC

Una vez que comienza el envío del streaming de audio por parte del terminal móvil, es necesario que en el receptor de la comunicación, esto es, el servidor de subtitulado, disponga de algún software que sea capaz de tratar este flujo continuo de datos.

Más en concreto, este software debe ser capaz de:

- Recibir tráfico UDP/RTP por un puerto específico.
- Reproducir audio codificado en PCM y AMR-NB en múltiples formatos.

VLC es un reproductor y framework multimedia del proyecto VideoLAN que soporta varios protocolos de comunicación, especialmente aquellos destinados a comunicaciones multimedia, así como multitud de formatos y códecs de audio y video. Además, es uno de los reproductores más independientes en cuanto a plataforma se refiere, con versiones para GNU/Linux, Microsoft Windows, Mac OS X y BSD, entre otros según su página oficial.

Después de realizar varias pruebas y consultar documentación precisa de VLC, se ha conseguido reproducir con este reproductor tráfico UDP y RTP por un puerto concreto cuyo contenido multimedia era audio codificado en PCM o AMR con diferentes configuraciones.

11.2.1.3.1 Requisitos

Es necesario disponer de la última versión del VLC, la cual se puede obtener a través de su web oficial de forma gratuita:

http://www.videolan.org/vlc/

No obstante, se proporciona el fichero de instalación (“vlc-2.2.X-win32.exe”) en el DVD.

Los requisitos para poder ejecutar correctamente VLC en el servidor de subtitulado son menores en cualquier caso que los necesarios para poder usar DNS 10.1, por lo que puede consultarse de nuevo la sección 11.2.1.2.1.

11.2.1.3.2 Instalación

El usuario puede ejecutar el fichero de instalación del VLC que se encuentra en el DVD e instalar el programa como cualquier otro habitual en un sistema operativo Windows.

11.2.2 Configuración

11.2.2.1 Configuración del dispositivo móvil: StreamDroid

11.2.2.1.1 Introducción

Una vez que StreamDroid se ha instalado correctamente en el smartphone, el usuario puede lanzar la aplicación desde la lista de aplicaciones. A continuación, aparece visible la pantalla principal de la aplicación (ver figura 67):

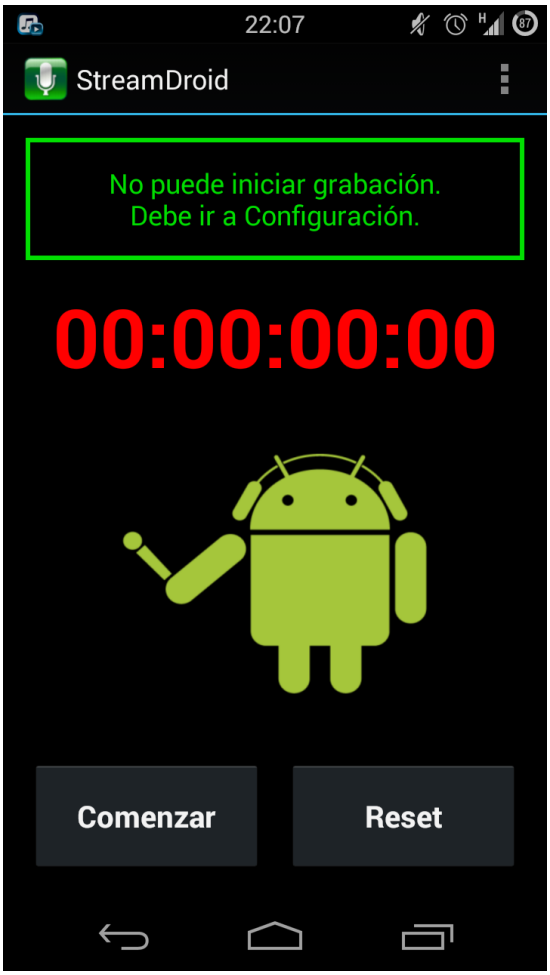

Figura 67. Pantalla principal de StreamDroid.

Como se puede apreciar en la anterior figura, hay tres secciones en la pantalla, además de la imagen central, para proporcionar información o interaccionar con el usuario:

- **Información del audio**: se indica el códec utilizado para la codificación de la voz (PCM/AMR), así como el protocolo de transporte para el streaming correspondiente (UDP/RTP).
  Inicialmente, tal y como se visualiza en la figura 67, no se proporciona ninguna información al respecto pues se asume que el usuario debe configurar sus preferencias al usar la aplicación por primera vez.

- **Temporizador**: cronómetro para indicar al usuario el tiempo que ha transcurrido de grabación de la voz. Tiene una precisión de centésimas de segundo. El color de los números en la pantalla varía en función del estado de la grabación: amarillo mientras la grabación de voz esta activa, rojo en cualquier otro caso.
- **Control de grabación**: se dispone de dos botones para controlar la grabación; uno para comenzar, pausar y reanudar la grabación y el otro para finalizar la grabación y resetear el temporizador. En cuanto se pulsa el botón correspondiente para comenzar a grabar la voz del usuario, automáticamente también empieza la transmisión del audio hacia el servidor siempre y cuando se haya configurado previamente.

La comunicación entre el terminal móvil y el servidor de subtitulado, al margen de los protocolos de red empleados en el transporte, puede ser a través de la red de datos móvil del propio terminal o mediante una red WiFi disponible a la que haya posibilidad de conexión.

Antes de comenzar la grabación del audio, es necesario previamente acceder a la pantalla de configuración para introducir algunos datos. Para ello, el usuario debe acceder a la opción "CONFIGURACION" a través del menú de la aplicación (ver figura 68).

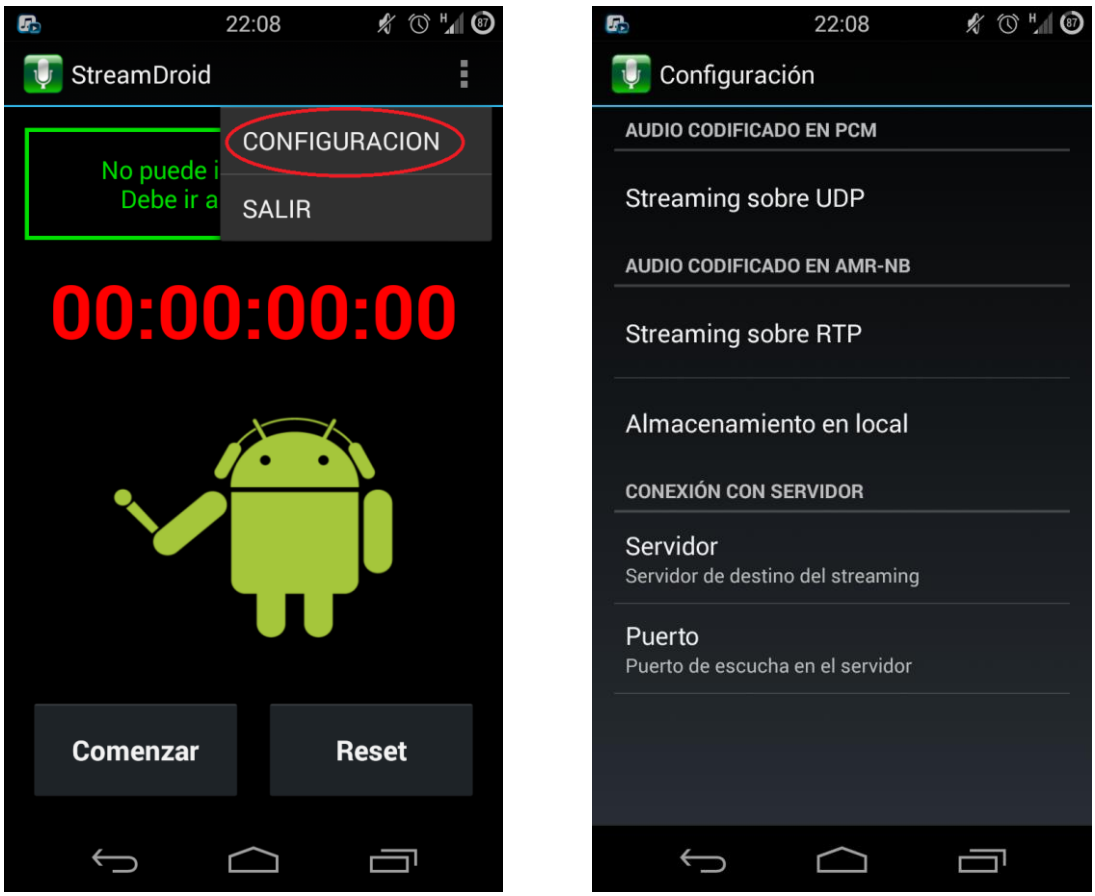

Figura 68. Acceso a la Configuración de StreamDroid.

La pantalla de configuración está dividida en tres secciones en función de las preferencias del tipo de códec que el usuario decida utilizar para la codificación de su voz (Audio codificado en PCM y Audio codificado en AMR-NB respectivamente) y de la conectividad con el servidor de subtitulado (ver figura 68).

11.2.2.1.2 Audio codificado en PCM

PCM es un procedimiento de modulación usado para transformar una señal analógica en una señal digital sin comprimir aportando la mayor calidad posible. Para el caso de StreamDroid, el audio PCM elegido presenta una codificación de 16 bits por muestra.

Por otro lado, el protocolo de transporte seleccionado para este tipo de formato en StreamDroid es UDP, ideal para cualquier transmisión de datos por streaming ya que no introduce retardos para establecer una conexión y la comunicación es unidireccional entre emisor (smartphone) y receptor (servidor de subtitulado).

Para habilitar el uso de audio PCM sobre un flujo UDP, el usuario debe activar dicha funcionalidad en la opción correspondiente de la pantalla de configuración (ver figura 69).

Figura 69 . Configuración Audio codificado en PCM para StreamDroid.

Una vez que se habilita la opción, es posible acceder a otras dos opciones:

- *Número de canales*. Puede ser mono (1 canal) o estéreo (2 canales).
- *Frecuencia de muestreo*. Posibles valores de 8000 y 16000 Hz (Herzios).

La máxima calidad de la voz se obtiene si se selecciona audio en estéreo y 16 KHz, pero sólo se aconseja su uso con redes de datos que proporcionen tasas altas de velocidad.

11.2.2.1.3 Audio codificado en AMR-NB

AMR-NB (en adelante solo AMR) es un estándar de codificación de audio muy usado para comunicaciones de voz en las redes móviles. Su principal característica es que puede gestionar de forma dinámica el ancho de banda al elegir entre ocho diferentes tasas de bits: 12.2, 10.2, 7.95, 7.40, 6.70, 5.90, 5.15 y 4.75 Kbps.

Su formato de compresión es muy superior al proporcionado por PCM, por tanto es más recomendable para entornos de conectividad donde las tasas de velocidad sean menores. Sin embargo, la calidad de la voz es menor.

Independientemente de la tasa de codificación que finalmente se decida utilizar, StreamDroid ofrece la posibilidad de elegir dos modos de grabación para el caso de la voz en AMR.

Grabación y envío de audio AMR por streaming RTP

El audio que se recoge del micrófono se codifica en AMR y se envía sobre un protocolo de nivel de sesión muy común para comunicaciones multimedia en tiempo real: RTP.

RTP es un protocolo que funciona por encima de UDP, el protocolo de transporte que usaba StreamDroid con el audio PCM, añadiendo un nivel de inteligencia y control a las transmisiones en tiempo real como VoIP (voz sobre IP).

En la figura 70 puede verse el indicador de la pantalla de configuración que habilita este modo.

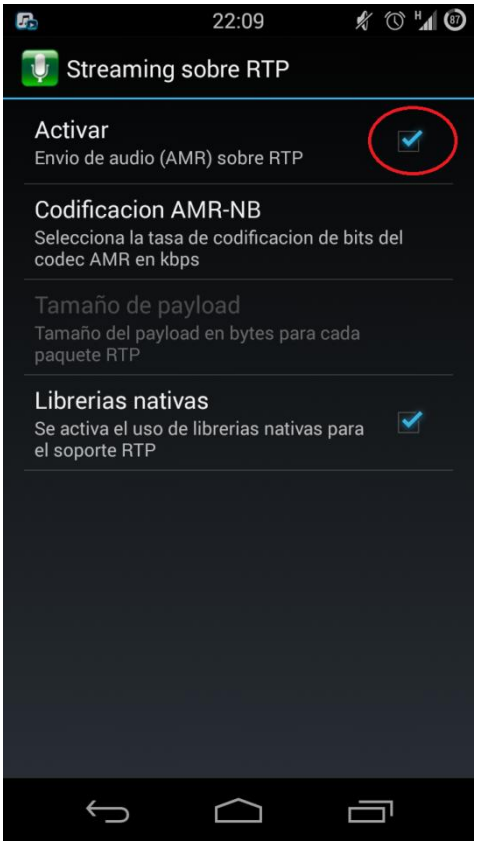

Figura 70 . Configuración Audio codificado en AMR para streaming.

Una vez activado el streaming de audio AMR por RTP, se posibilita el acceso a tres opciones para configurar este modo de funcionamiento:

- *Tasa de codificación*. Una de las 8 codificaciones disponibles en Kbps. Por defecto, se establece a 12.2 Kbps, la de mayor calidad.
- *Tamaño de payload*. Tamaño en bytes de los datos de audio codificados (carga útil) en cada paquete RTP. Por defecto, se establece en 500 bytes. Únicamente tiene sentido si no se usan las librerías nativas de Android para soporte RTP.
- *Librerías nativas*. Si se activa esta funcionalidad, la aplicación hace uso de las librerías nativas para poder realizar streaming RTP, las cuales únicamente están disponibles a partir de Android 3.1. Por tanto, los dispositivos con versiones anteriores de Android no podrán habilitar esta funcionalidad y el soporte a RTP se realizará a través de librerías de terceros.

Grabación de audio AMR en memoria interna

Consiste en la grabación del flujo de audio en un fichero (con extensión AMR) que se almacena de forma local en el smartphone y que no se transmite en tiempo real a ningún servidor; es decir, no se realiza streaming a ningún destino.

Para habilitar dicho modo de funcionamiento, el usuario debe habilitar la opción que corresponda en la pantalla de configuración (ver figura 71). Tras este cambio, es posible acceder a otras tres opciones:

Figura 71 . Configuración Audio codificado en AMR para almacenamiento.

Esta modalidad resulta muy útil cuando no existe la posibilidad de conexión a Internet, puesto que todo el audio grabado queda almacenado y puede extraerse y analizarse posteriormente.

- *Tasa de codificación*. Una de las 8 codificaciones disponibles en Kbps. Por defecto, se establece a 12.2 Kbps, la de mayor calidad.
- *Nombre directorio*. El nombre del directorio donde se almacenarán los ficheros de audio. Dicho directorio se ubicará en la raíz de la unidad primaria de almacenamiento del dispositivo.
- *Nombre fichero*. El nombre del fichero con extensión 'amr'. Si no se especifica la extensión, StreamDroid la añade automáticamente.

11.2.2.1.4 Conexión con servidor

Finalmente es necesario especificar la configuración con el servidor de subtitulado en términos de conectividad, pues en caso contrario jamás se podría transportar el flujo de audio y no existiría el concepto de streaming.

- *Servidor*. El servidor de destino (idealmente, el servidor de subtitulado) mediante una dirección IP o su nombre de host.
- *Puerto*. El puerto UDP desde el cual el servidor recibe los datos.

11.2.2.1.5 Inicio y control de grabación

En caso de haber finalizado la configuración de StreamDroid con éxito, el usuario puede volver a la pantalla principal y pulsar el botón "Comenzar" para iniciar la grabación de su voz. En ocasiones puede producirse un pequeño retardo en el inicio de la grabación derivado de configuraciones previas y comprobaciones iniciales. También puede darse el caso de que la configuración elegida por el usuario no sea correcta o que no haya conectividad de datos, en cuyo caso no se inicia la grabación y se notifica con una ventana emergente.

El aspecto que presenta la pantalla principal para cada uno de los formatos de audio transcurridos varios segundos aproximadamente desde el inicio de la grabación se puede ver tanto en la figura 72 como en las figura 73.

Tanto la opción "Pausar" como la de "Reset" finalizan la grabación y envío de audio pero únicamente la de "Reset" reinicia el contador del tiempo. El mismo efecto causa en la grabación en curso si el usuario decide modificar alguna de las preferencias de la aplicación al acceder a la pantalla de configuración.

11.2.2.1.6 Finalización de aplicación

Para cerrar la aplicación desde la pantalla principal se puede seleccionar la opción "*SALIR*", a la cual se accede de la misma forma que a la sección de configuración (ver figura 68). Al elegir esta opción, automáticamente se finalizan todas las tareas en curso de la aplicación y se mata el proceso de sistema asociado a StreamDroid.

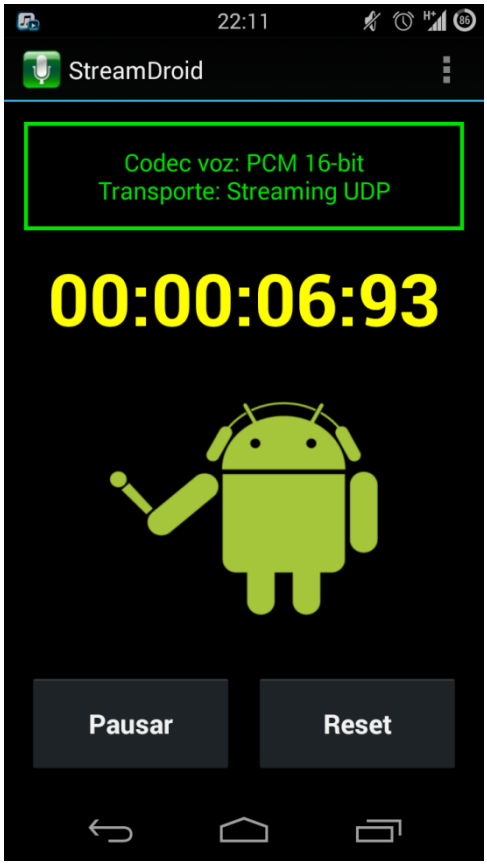

Figura 72. Pantalla principal de StreamDroid durante la grabación y envío de audio PCM.

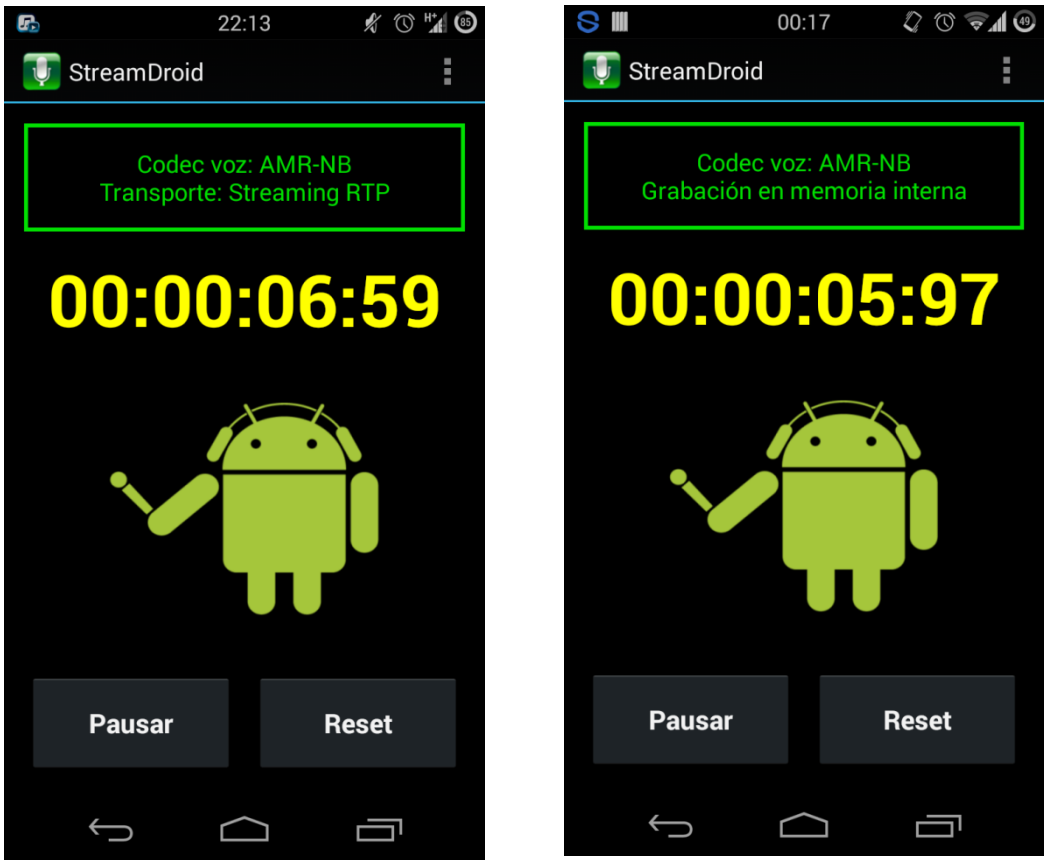

Figura 73. Pantalla principal de StreamDroid durante la grabación y envío de audio ARM.

11.2.2.2 Configuración del Servidor de Reconocimiento de Voz

11.2.2.2.1 Reproducción del streaming con VLC

Una vez que la aplicación en el smartphone se encuentra configurada y operativa, el siguiente paso es conseguir que el flujo de audio se reproduzca correctamente en el servidor de subtitulado.

Independientemente de la configuración que haya realizado el usuario para StreamDroid, cualquiera de los flujos de audio recibidos se pueden reproducir con el reproductor VLC.

Por ello, y asumiendo que se tiene instalada la versión 2.2.X de VLC, hay que lanzar dicha aplicación y seguir uno de los dos siguientes procedimientos en función de la naturaleza del audio generado por StreamDroid:

Reproducción de audio PCM sobre UDP

Una vez que se tiene en primer plano la pantalla principal de VLC, se debe acceder a la pantalla de configuración de recursos de red a través de la siguiente ruta (ver figura 74):

Medio → Abrir ubicación de red

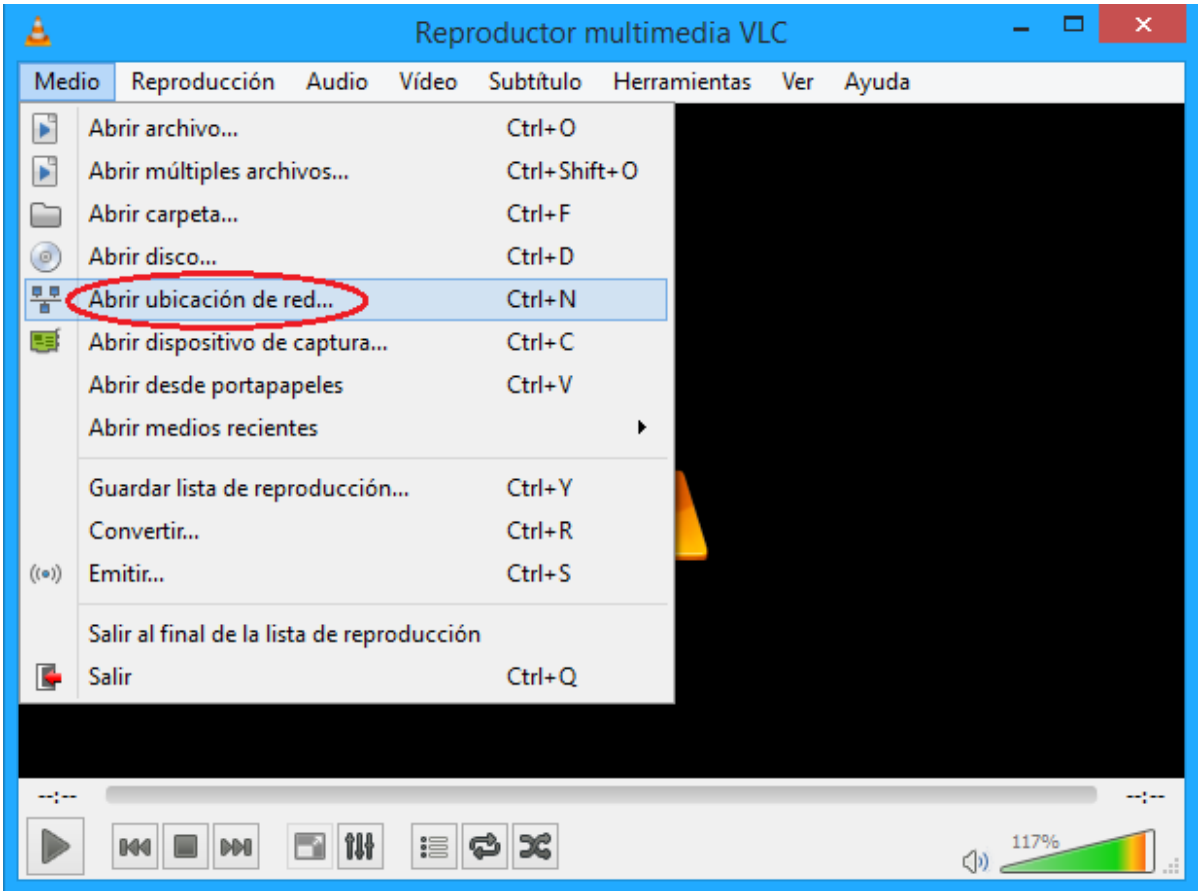

Figura 74 . Acceso a Configuración de recursos de red en VLC.

En la pantalla de configuración de recursos de red (ver figura 75) hay que especificar en primera instancia el puerto de escucha UDP en el servidor. Para ello, en la opción "Introducir una URL" se inserta la siguiente instrucción:

```
udp://@:XXXXX
```

Siendo *XXXXX* el puerto de escucha.

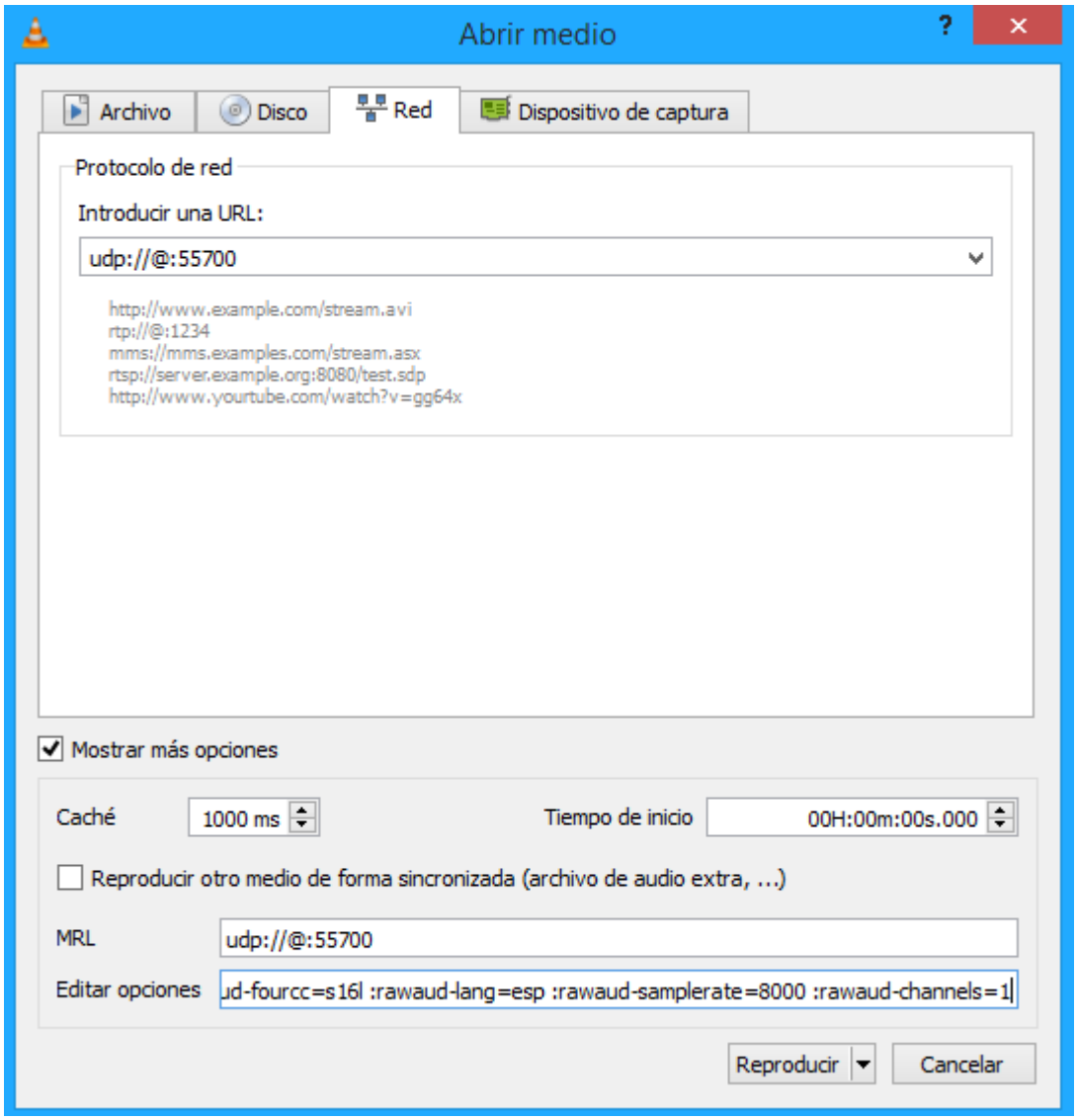

Figura 75 . Pantalla de Configuración de recursos de red en VLC.

A continuación se debe habilitar la opción "Mostrar más opciones" e introducir los parámetros propios de la configuración del audio en la opción "Editar opciones". Un ejemplo de dicha configuración podría ser la siguiente:

```
demux=rawaud :rawaud-fourcc=s16l :rawaud-samplerate=8000 :rawaud-channels=1
```

La especificación de estos parámetros en el momento de ejecutar VLC se debe a que en StreamDroid se eligieron estos valores para la codificación de la voz en PCM:

- rawaud-fourcc=s16l → 16 bits por muestra.
- rawaud-samplerate=8000 → frecuencia de muestreo de 8 KHz (también sería válido el valor 16000 para 16 KHz).
- rawaud-channels=1 → un solo canal o mono (también sería válido el valor 2 para sistema estéreo o de dos canales).

Finalmente hay que pulsar el botón "Reproducir" y en el servidor de subtitulado aparecerá la pantalla habitual del VLC con la reproducción en curso del streaming (ver figura 76), suponiendo que StreamDroid se encuentre en uso y emitiendo audio.

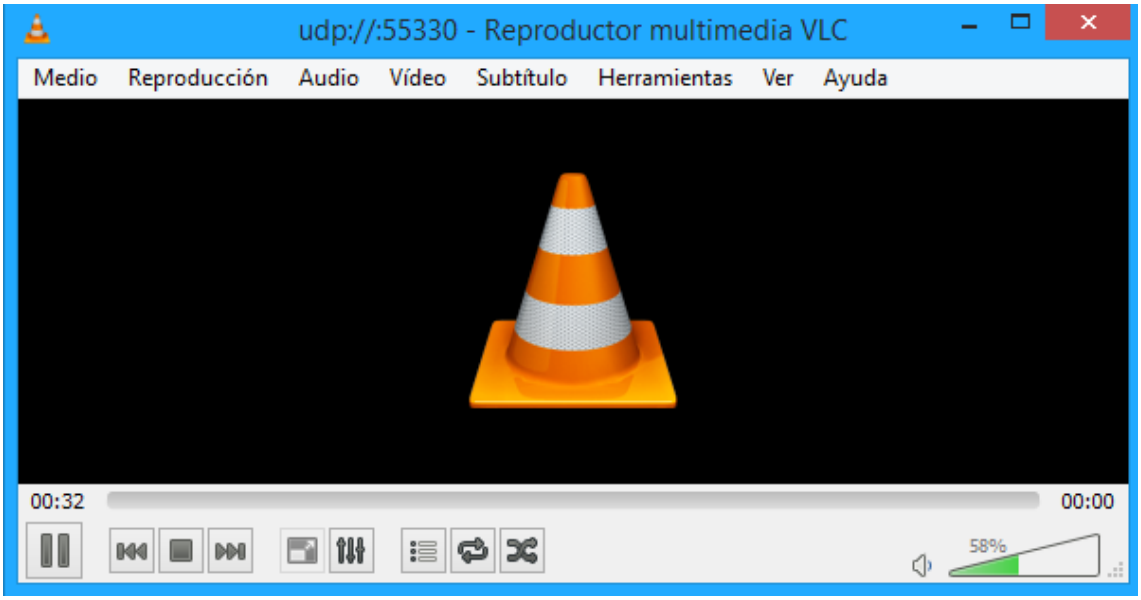

Figura 76. Ventana de reproducción de VLC para audio PCM sobre UDP.

Reproducción de audio AMR sobre RTP

Si en lugar de audio PCM, StreamDroid se configura para usar voz codificada en ARM sobre RTP, se puede utilizar de nuevo VLC para reproducir el streaming en el servidor, pero el procedimiento para ello es diferente.

En primer lugar, se debe construir un fichero de texto plano que permita indicar la configuración de audio mediante SDP (Session Description Protocol), un protocolo diseñado para describir los parámetros de inicialización de los flujos multimedia.

D esta forma, SDP se encargaría de entablar una negociación entre el VLC y StreamDroid para definir el tipo de contenido, formato, y todos los demás parámetros necesarios para formalizar la sesión.

El fichero de texto que definiría dicha sesión se describe con una serie de atributos, representados cada uno de ellos con un solo carácter seguido por '=' y el valor (puede ser más de uno) correspondiente. Cada atributo se encuentra especificado en una línea.

La extensión del fichero debe ser *sdp*, pudiéndose renombrar con el nombre que se quiera.

Para el caso de audio AMR sobre RTP, las líneas adecuadas serían las siguientes:

```
v=0

o=StreamDroid 0 0 IN IP4 127.0.0.1

s=PFC Alvaro Martin

c=IN IP4 XXX.XXX.XXX.XXX

t=0 0

m=audio YYYYY RTP/AVP 97

a=rtpmap:97 AMR/8000/1

a=fmtp:97 octet-align=1;mode-set=0,1,2,3,4,5,6,7
```

Únicamente se debe modificar el valor de dos atributos para completar la configuración, es decir, dos de las líneas; el resto de los atributos han de mantener la misma estructura y valores que los indicados:

- *'c':* Información de la conexión.

  XXX.XXX.XXX.XXX representa la dirección IP del servidor. Es opcional, se puede prescindir de esta especificación, pero se recomienda su uso.

- *'m':* Nombre de medio y dirección de transporte.

  YYYYY indica el puerto de escucha. Es necesario especificar este parámetro.

Una vez que el fichero SDP se encuentra configurado y disponible, y asumiendo que se tiene en primer plano la pantalla principal de VLC, se debe acceder a la siguiente ruta (ver figura 77):

Medio → Abrir archivo…

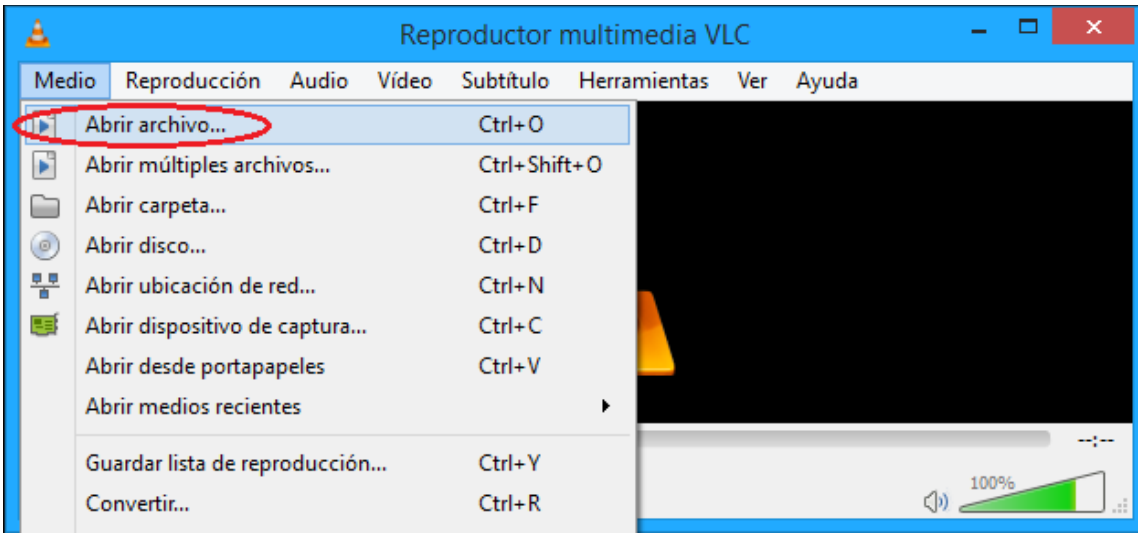

Figura 77 . Acceso a Selección de archivo en VLC.

Por último, hay que elegir el archivo SDP correspondiente y aparecerá la pantalla habitual del VLC con la reproducción del streaming de formar muy similar a lo mostrado ya en la figura 76.

11.2.2.2.2 Conexión con DNS

Suponiendo que se ha conseguido reproducir cualquiera de los diferentes tipos de streaming de voz con el reproductor VLC, se precisa en cualquier caso conseguir que el flujo de voz sea la fuente de audio para DNS.

Hay que recordar que la versión de DNS elegida no permite recoger el audio a transcribir directamente de la tarjeta de red (se recuerda que el audio procede del terminal móvil y se ha enviado por streaming a través de Internet). Más en concreto, DNS solo permite especificar como fuente de audio para la transcripción la entrada de una tarjeta de sonido.

Para solventar este problema y lograr el objetivo, el sistema operativo del servidor de subtitulado debe procesar la salida del VLC y enviar de forma constante y progresiva datos válidos hacia la entrada de audio del sistema; es decir, hay que conseguir conectar la salida de la tarjeta de sonido a su propia entrada, realimentando de esta manera el flujo de audio en este dispositivo.

La solución viable elegida, tanto por su bajo coste como por simplicidad, es la utilización de un único cable de 2 conectores macho Jack de 3.5 mm (ver figura 78) en cada extremo para unir de forma directa la salida a la entrada de la tarjeta de sonido. La mayoría de las tarjetas de sonido actuales soportan esta posibilidad, independientemente del sistema operativo que las controle.

Figura 78 . Cable conectores Jack 3.5 mm.

Asumiendo una situación en la que el usuario se encuentre enviando un flujo constante de voz desde un smartphone con StreamDroid hasta un servidor donde una instancia de VLC se haya configurado parar escuchar y reproducir dicho flujo y un cable como el de la figura 78 conecte las interfaces correspondientes de la tarjeta de sonido, es necesario verificar que el sistema recibe un buen nivel de señal.

Para dicha comprobación, se debe acudir a la configuración del sonido del sistema desde el panel de control de Windows y una vez en esta pantalla acceder a la pestaña “Grabar” tal y como se muestra en la figura 79.

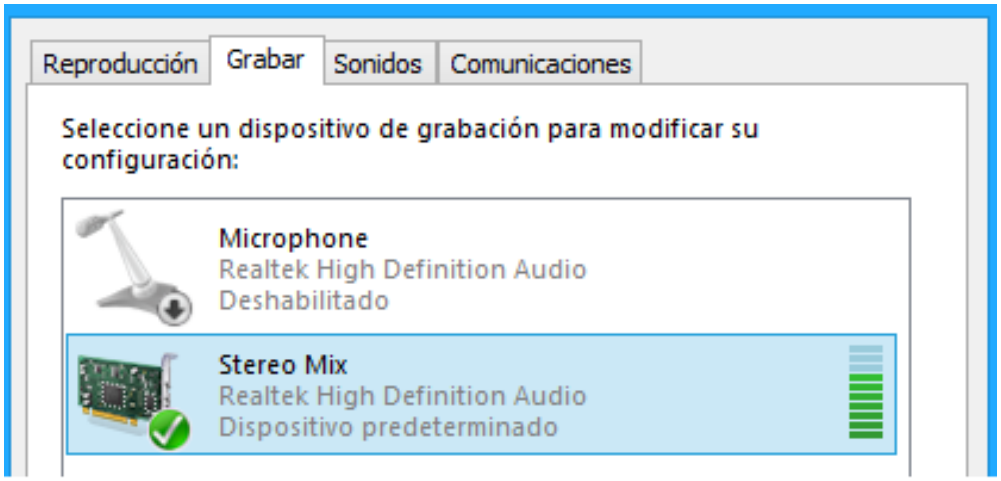

Figura 79. Nivel de señal en el dispositivo de grabación del sistema.

Se puede ajustar el nivel de señal si no es el adecuado o el deseado accediendo a las propiedades del dispositivo de grabación concreto (ver figura 80).

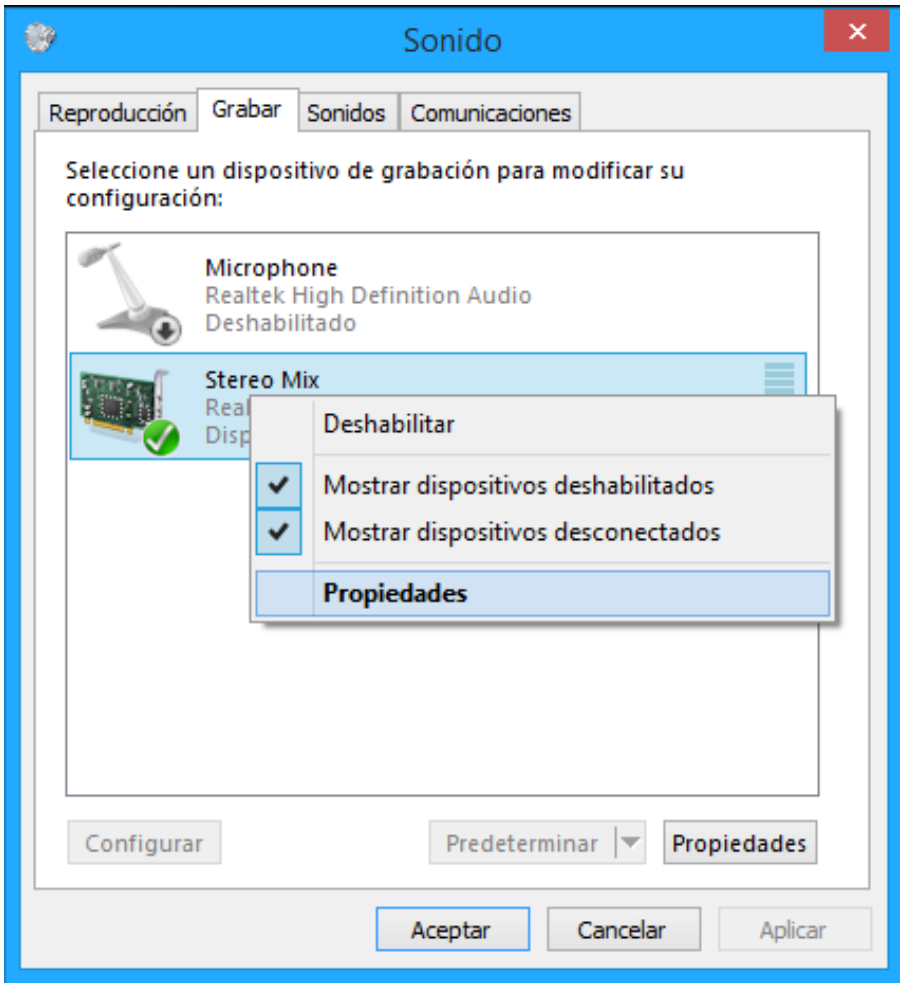

Figura 80. Acceso a las propiedades del dispositivo de grabación del sistema.

Una vez en la pantalla de propiedades, ya es posible seleccionar de forma manual el nivel e incluso acceder a opciones relacionadas más avanzadas (ver figura 81).

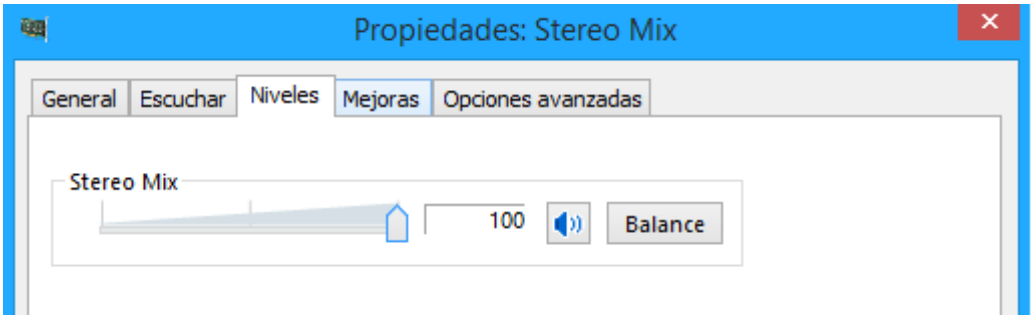

Figura 81. Modificación del nivel de señal del dispositivo de grabación del sistema.

Sin embargo, existen también otras alternativas para conseguir la funcionalidad requerida.

Alternativa con dos tarjetas físicas de sonido

La primera alternativa que se ha investigado es la utilización de un servidor de subtitulado con dos tarjetas físicas de sonido disponibles con las siguientes características:

- Ambas deben tener la misma interfaz para no perder calidad de audio en la transmisión de la señal de voz entre ellas.
- Cable de conexión entre las interfaces de cada tarjeta (ver figura 78).

Con esta arquitectura se consigue enviar por la salida de la primera tarjeta el audio digital a partir del flujo continuo que VLC consigue reproducir por el puerto UDP en el que se mantiene en escucha. Por otro lado, dicha salida se conecta con la entrada de la segunda tarjeta de forma directa mediante cableado físico, de manera que el audio digital se mapea directamente como entrada del DNS a través de la segunda tarjeta.

Por tanto, es imprescindible el uso de dos tarjetas de sonido: la primera para interpretar la salida del VLC y la segunda, para especificar a DNS que utilice como fuente de audio su entrada, que será a su vez, la salida de la primera tarjeta.

Alternativa con tarjetas virtuales de sonido

Otra solución válida para la limitación de DNS para no poder procesar directamente el audio recibido por streaming es instalar una tarjeta virtual de sonido en el servidor de subtitulado y utilizar ésta en las mismas condiciones que la segunda tarjeta física del anterior caso.

Existen varias soluciones de tarjetas virtuales como las siguientes:

- http://www.virtualaudiostreaming.net/
- http://virtualsoundcarddriver.com/
- https://www.audinate.com/products/software/dante-virtual-soundcard
- http://sourceforge.net/projects/vdmsound/

Se ha realizado una pequeña prueba de concepto de esta parte utilizando una tarjeta de sonido virtual con resultado satisfactorio.

11.2.2.2.3 Configuración de DNS

Una vez que se ha logrado enviar el flujo de audio a la entrada de alguna tarjeta de sonido del servidor de subtitulado por cualquiera de los métodos explicados en el anterior apartado, se debe configurar el software de transcripción a texto: DNS.

Tras arrancar la herramienta, se debe crear un nuevo perfil de usuario. Para ello, el usuario debe acceder desde el menú principal hasta el asistente de creación de usuarios para DNS:

Naturally Speaking → Administrar usuarios → Nuevo…

Se debe indicar un nombre de usuario (ver figura 82) e ir a la siguiente sección manteniendo el resto de opciones tal y como aparecen.

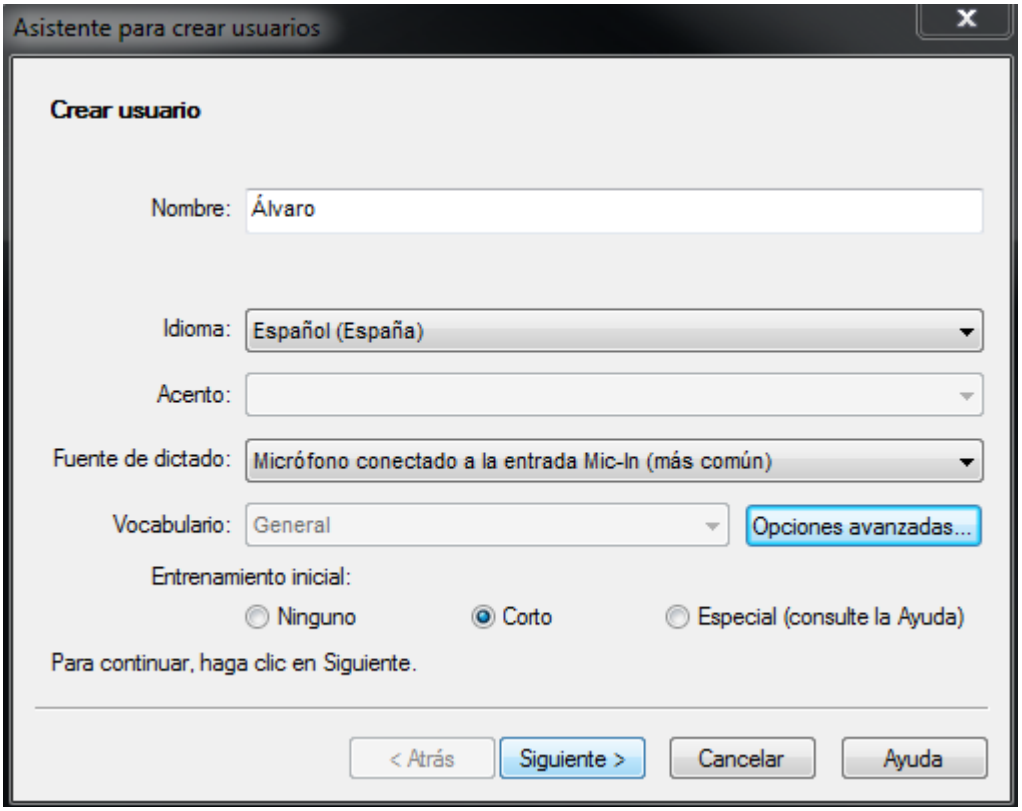

Figura 82. Asistente de creación de usuarios de DNS.

La siguiente pantalla sugiere al usuario recomendaciones para la colocación del micrófono, por lo que se puede aprovechar este momento para configurar StreamDroid y su conectividad con el servidor de subtitulado en caso de que no se hubiera hecho con anterioridad.

A continuación, DNS ofrece la posibilidad de calibrar el volumen tal y como se muestra en la figura 83. Para ello, se debe seleccionar “Iniciar prueba de volumen” y leer en voz alta el texto que aparece en pantalla con la aplicación StreamDroid. La barra azul situada a la derecha de la pantalla indica el volumen obtenido y puede fluctuar con regularidad durante la prueba.

Si se consigue obtener un volumen adecuado para poder realizar una correcta transcripción, DNS indica que el paso se ha completado (ver figura 84).

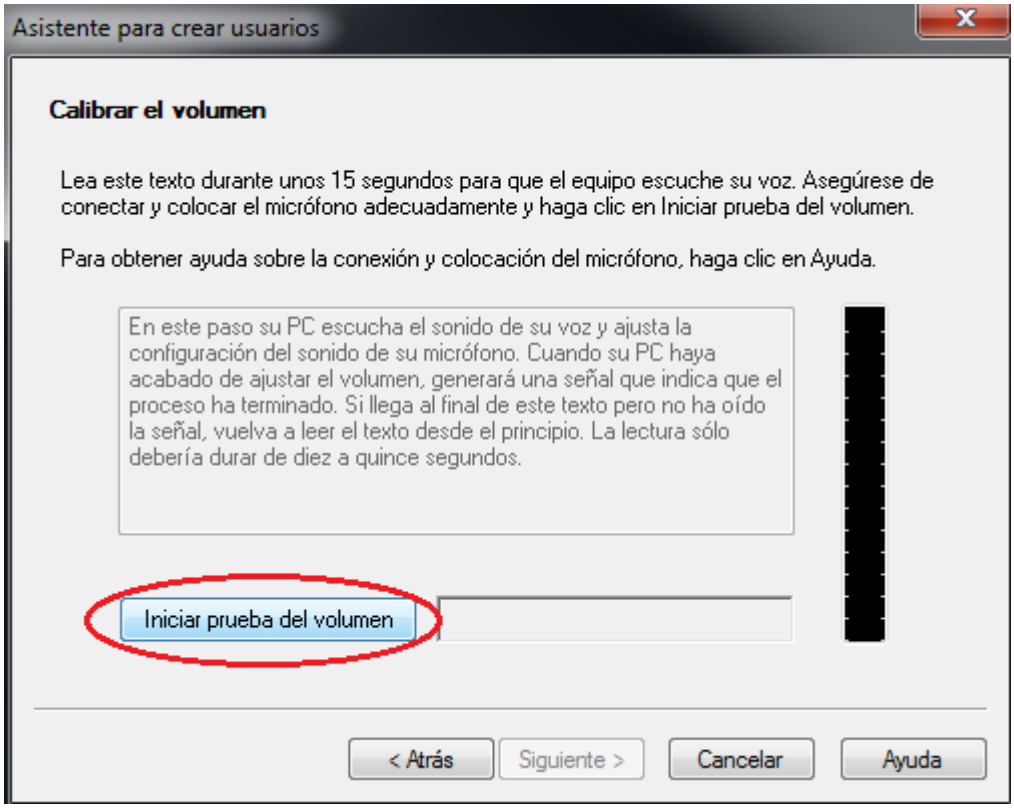

Figura 83. Inicio de prueba de volumen en DNS.

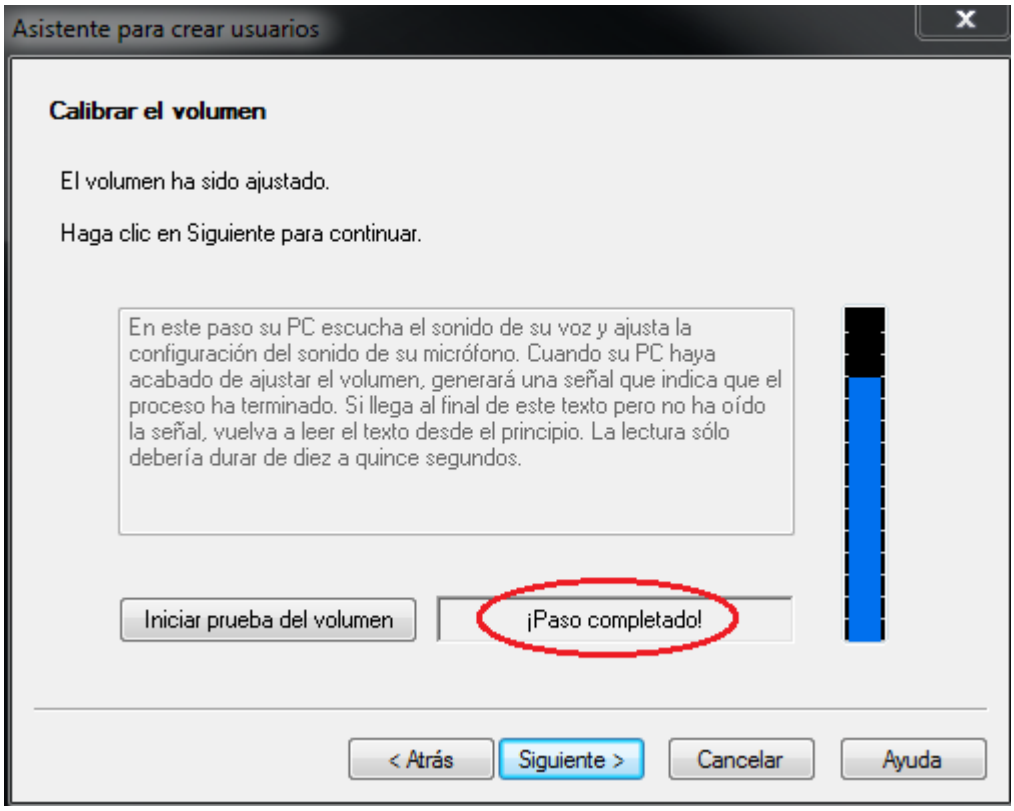

Figura 84. Resultado de prueba de volumen en DNS.

La siguiente sección consiste en una prueba de la calidad de sonido (ver figura 85) y, al igual que en el caso anterior, se debe leer en voz alta el texto que aparece en pantalla.

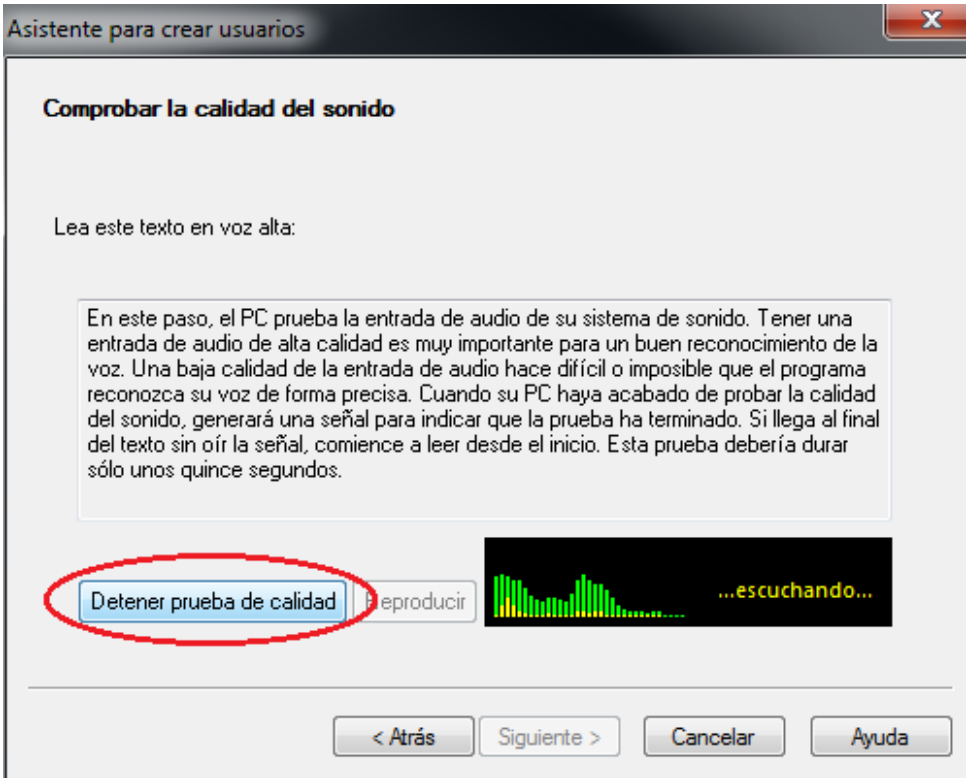

Figura 85. Inicio de prueba de calidad de sonido en DNS.

Una vez finalizada la lectura, DNS indica la relación voz-ruido (ver figura 86).

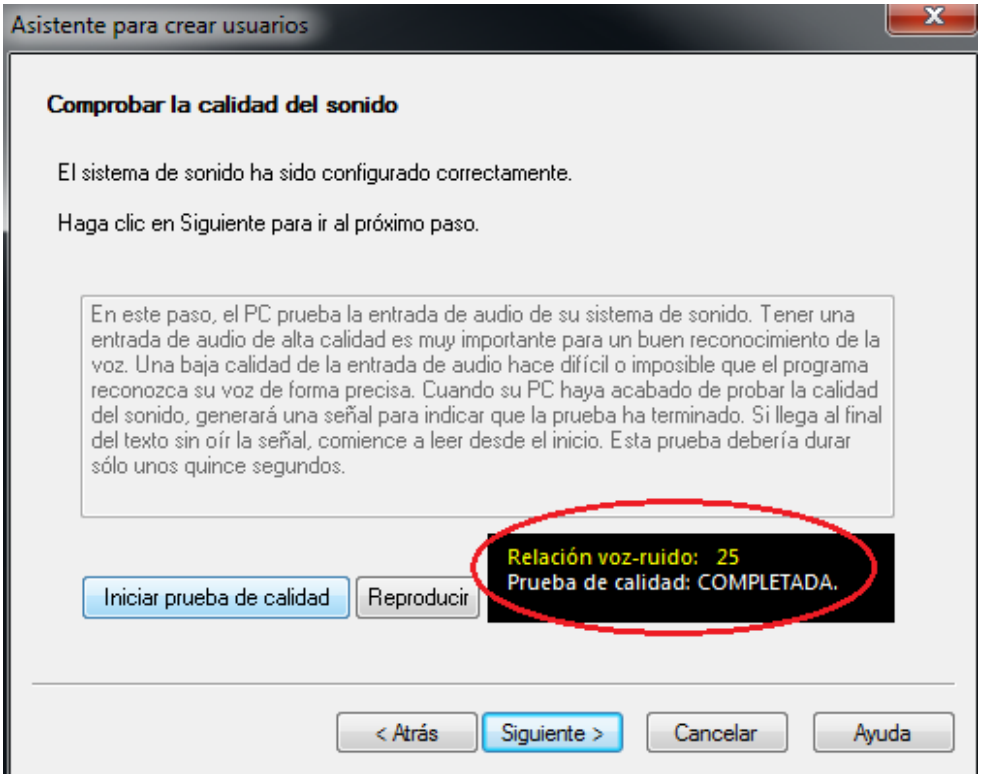

Figura 86. Resultado de prueba de calidad de sonido en DNS.

Además ofrece de las opciones de reproducir el audio del texto que se acaba de leer o de repetir la prueba de calidad. En cualquier caso, y directamente relacionado con el valor obtenido de la relación voz-ruido, DNS no permitirá continuar con el proceso si considera que no se ha obtenido un buen resultado.

Tanto la calibración del volumen como la prueba de calidad del sonido constituyen aspectos muy importantes de la configuración pues una correcta transcripción de sonido a texto depende directamente de la calidad e intensidad de la señal de voz registrada.

El paso siguiente en la configuración es el entrenamiento de DNS para el perfil de usuario que se está creando. La voz humana es una característica propia y única de cada individuo por lo que es imprescindible realizar al menos un entrenamiento inicial por cada perfil configurado ya que, en caso contrario, es muy probable que DNS no consiga reconocer correctamente los comandos de voz.

Tras acceder a la pantalla de inicio del entrenamiento de DNS (ver figura 87), se debe proceder a su arranque mediante la opción “Comenzar” y a continuación leer de forma clara y con pausa el texto que va apareciendo en pantalla.

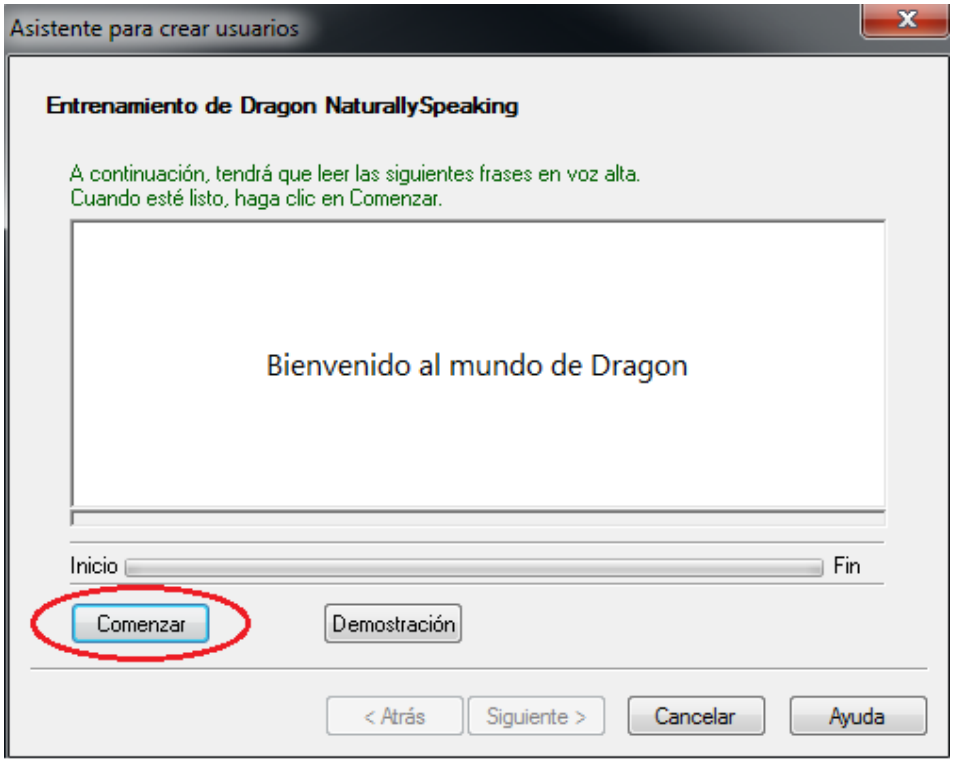

Figura 87. Inicio de entrenamiento de DNS.

Después de que una serie de frases sean leídas correctamente, DNS muestra al usuario una lista con títulos de textos largos, de los cuales se debe elegir únicamente uno de ellos para continuar con el entrenamiento (ver figura 88).

Una vez seleccionado el texto largo, se debe proseguir con su lectura de la forma más correcta y completa posible por parte del usuario. Al finalizar, DNS procesa y adapta todos los resultados obtenidos al perfil de usuario (ver figura 89).

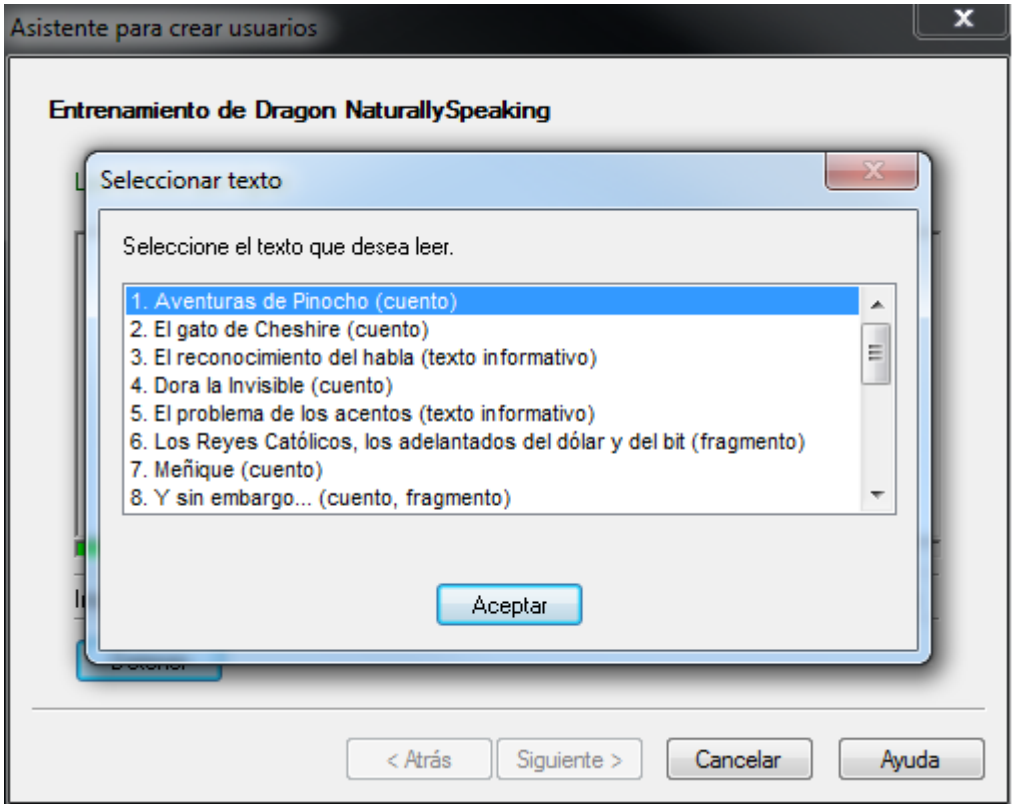

Figura 88. Lista de textos largos en DNS.

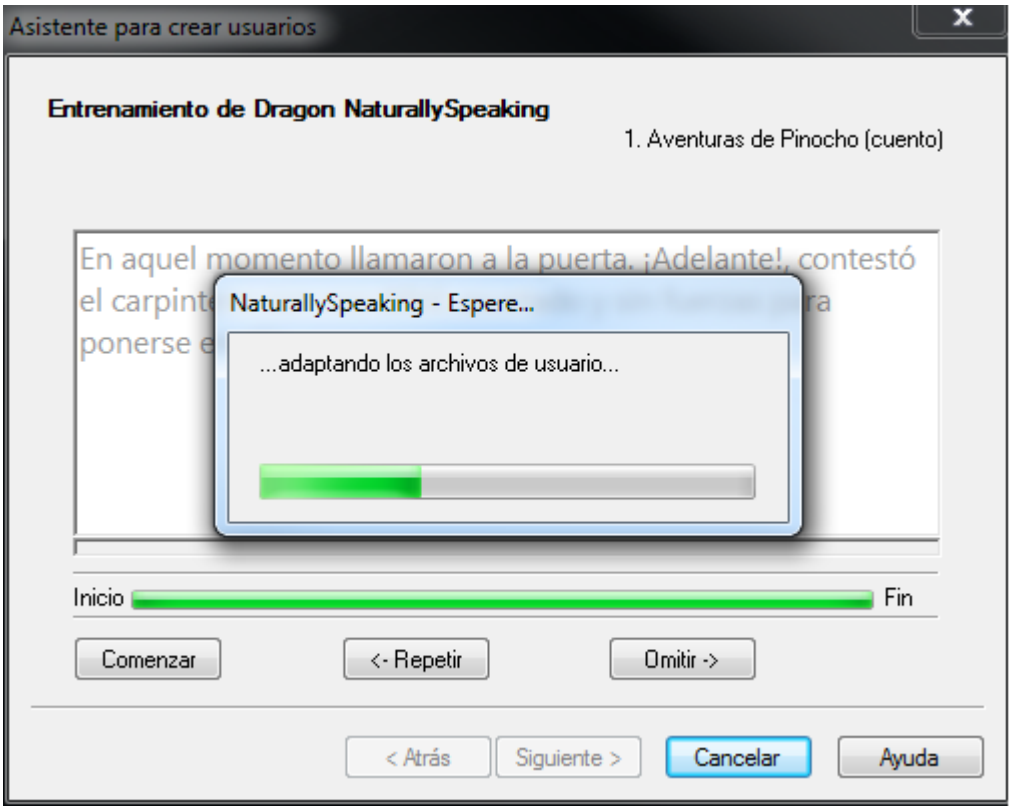

Figura 89. Fin de entrenamiento de DNS.

El último punto para completar la creación del perfil de usuario, y en general la configuración de DNS, son las opciones relacionadas con el reglaje de precisión y la recopilación de datos.

En ambos casos, el usuario debe desactivar dichas opciones tal y como aparece en la figura 90, puesto que no son imprescindibles para completar el proceso.

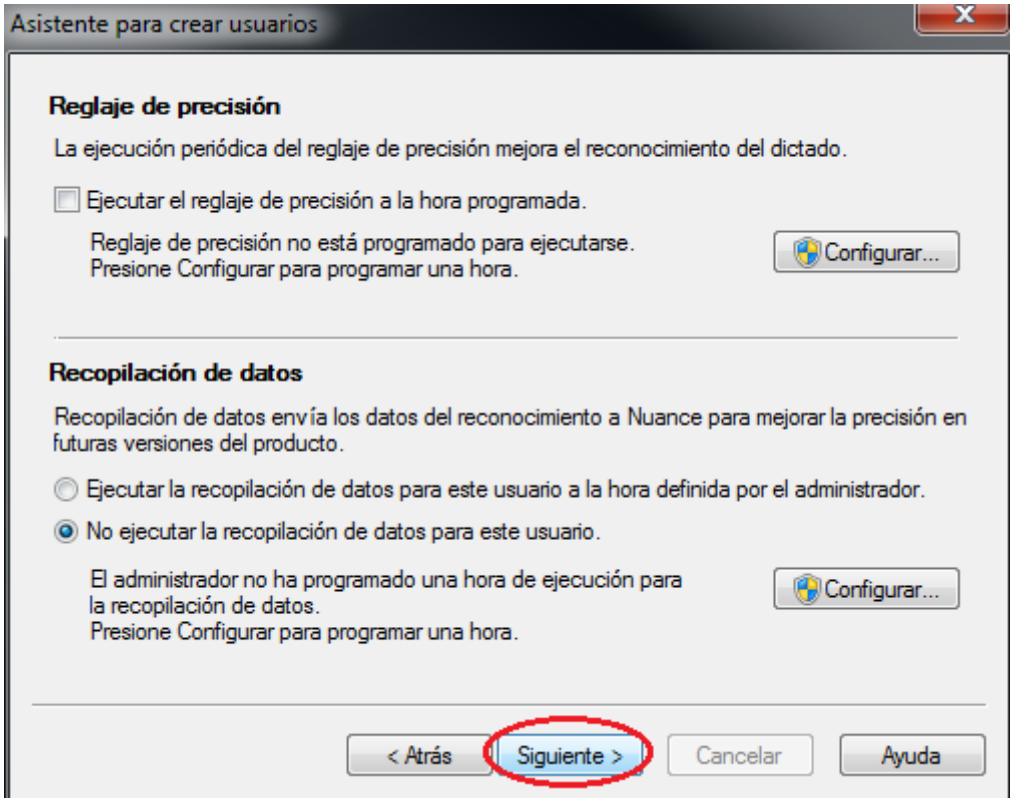

Figura 90. Últimas opciones en la creación de un usuario en DNS.

Por último, DNS indica que el proceso de creación de usuario ha finalizado correctamente y ofrece tres posibilidades antes de continuar:

1. Iniciar el Paseo rápido.
2. Novedades de la versión 10.
3. Empezar a dictar.

El usuario puede elegir de forma libre cualquiera de las tres, pero se recomienda la primera de ellas para aprender a manejar las funciones básicas de DNS.

A partir de este momento, se dispone de un perfil personal para DNS adaptado a la voz propia del usuario, que siempre se podrá volver a entrenar y editar si los resultados obtenidos en las primeras pruebas no son los esperados.

Por otro lado, se recomienda la creación de un perfil en DNS por cada usuario como ya se había comentado y además por cada uno de los dos tipos de formatos de audio que se manejan en el sistema (AMR-NB y PCM), dada la diferente calidad de audio que proporcionan cada uno de ellos.

Por ejemplo, si el sistema va a ser manejado por 3 personas se deberían disponer de al menos 6 perfiles de usuario. El proceso de creación de usuario y entrenamiento en DNS es idéntico en cualquier caso.

11.2.3 Primeras pruebas

Una vez que se ha finalizado con éxito el proceso de instalación y el de configuración de todo el sistema, se pueden comenzar a realizar las primeras pruebas. Para ello, con el sistema al completo funcionado, es decir, StreamDroid enviando un flujo de voz por streaming hasta el servidor de subtitulado y redirigiendo dicho flujo hasta la entrada de la tarjeta de sonido, se debe iniciar DNS y seleccionar uno de los perfiles de usuario.

Para acceder a la pantalla de selección de usuario en DNS, se puede proceder a través de la siguiente ruta (ver figura 91):

NaturallySpeaking → Abrir usuario

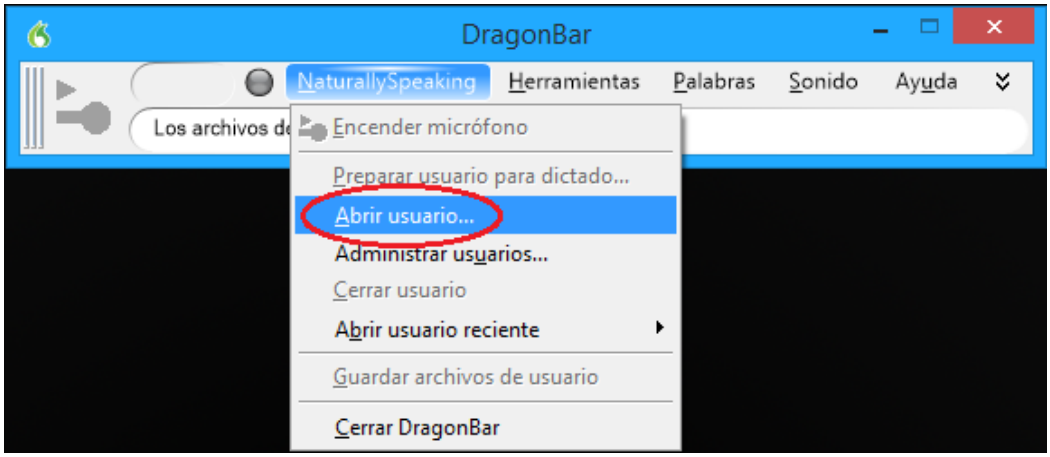

Figura 91. Acceso a pantalla de selección de usuario en DNS.

Y a continuación seleccionar uno de los perfiles creados (ver figura 92):

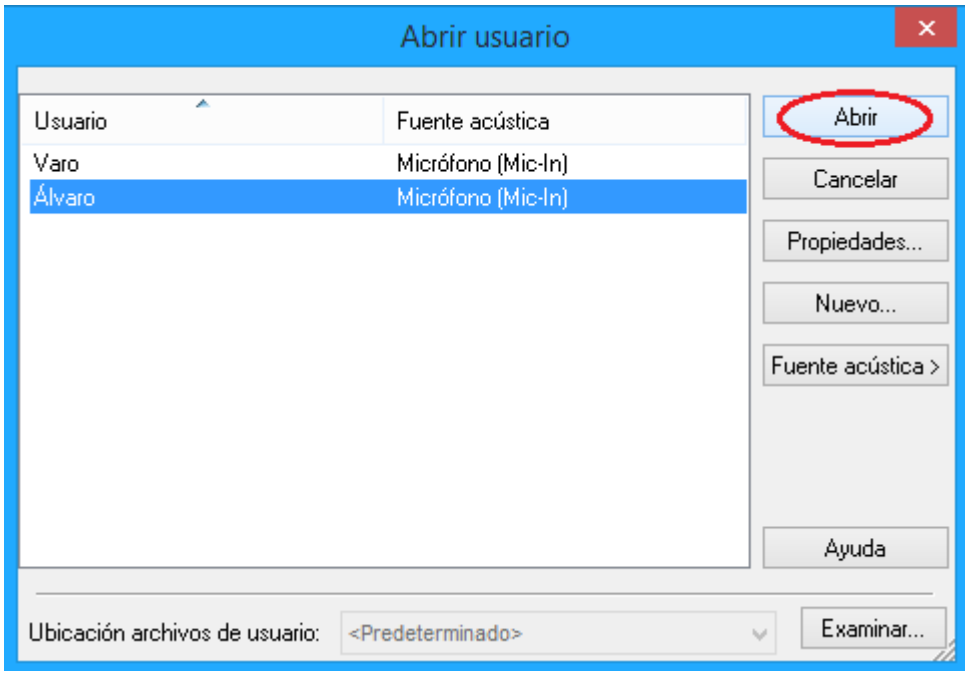

Figura 92. Pantalla de selección de usuario en DNS.

Tras la selección de un determinado usuario, DNS necesita realizar la carga del perfil en su sistema y, en muchas ocasiones, el proceso puede tardar varios segundos tal y como aparece a continuación en la figura 93:

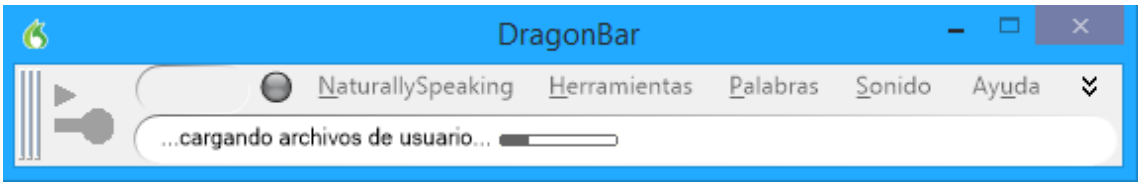

Figura 93. Proceso de carga de perfil de usuario en DNS.

Una vez finalizado el anterior proceso de carga de archivos, se puede arrancar alguna de las herramientas que facilita DNS para las transcripciones de audio a texto como el DragonPad: un editor y procesador de texto con un formato similar a las versiones primitivas de Microsoft Word, en el cual se pueden realizar dictados de voz en tiempo real.

Se puede acceder a esta herramienta desde la siguiente ruta en DNS (ver figura 94):

Herramientas → DragonPad.

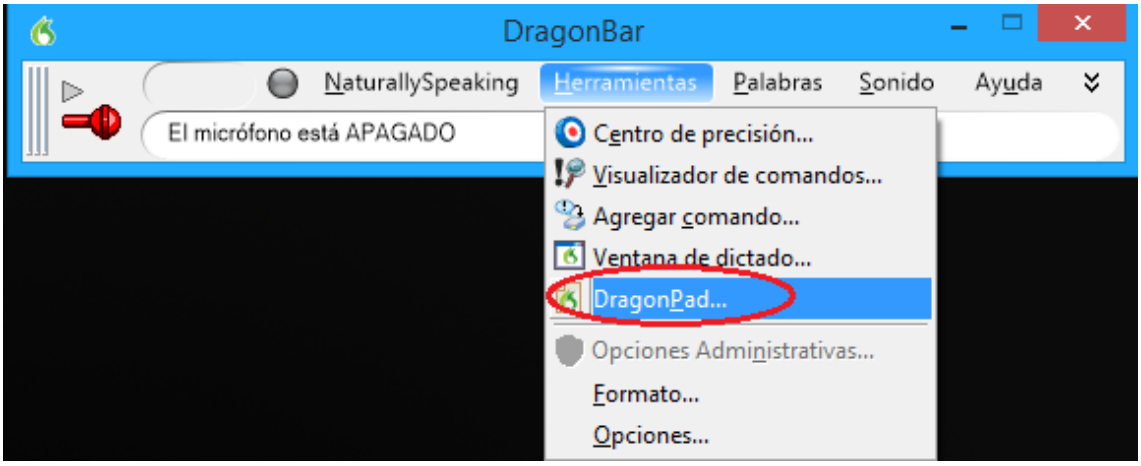

Figura 94. Acceso a la herramienta DragonPad de DNS.

El último paso antes de comenzar el dictado consiste en activar el micrófono de DNS para indicar al software que debe procesar el sonido desde la entrada de audio (ver figura 95).

Figura 95. Activación de micrófono en DNS.

Una vez activado el micrófono de DNS (color verde), puede iniciarse el dictado de voz manteniendo en un primer plano de la pantalla la herramienta DragonPad.

A medida que se realiza el dictado, el texto va apareciendo por pantalla tal y como se aprecia en la figura 96.

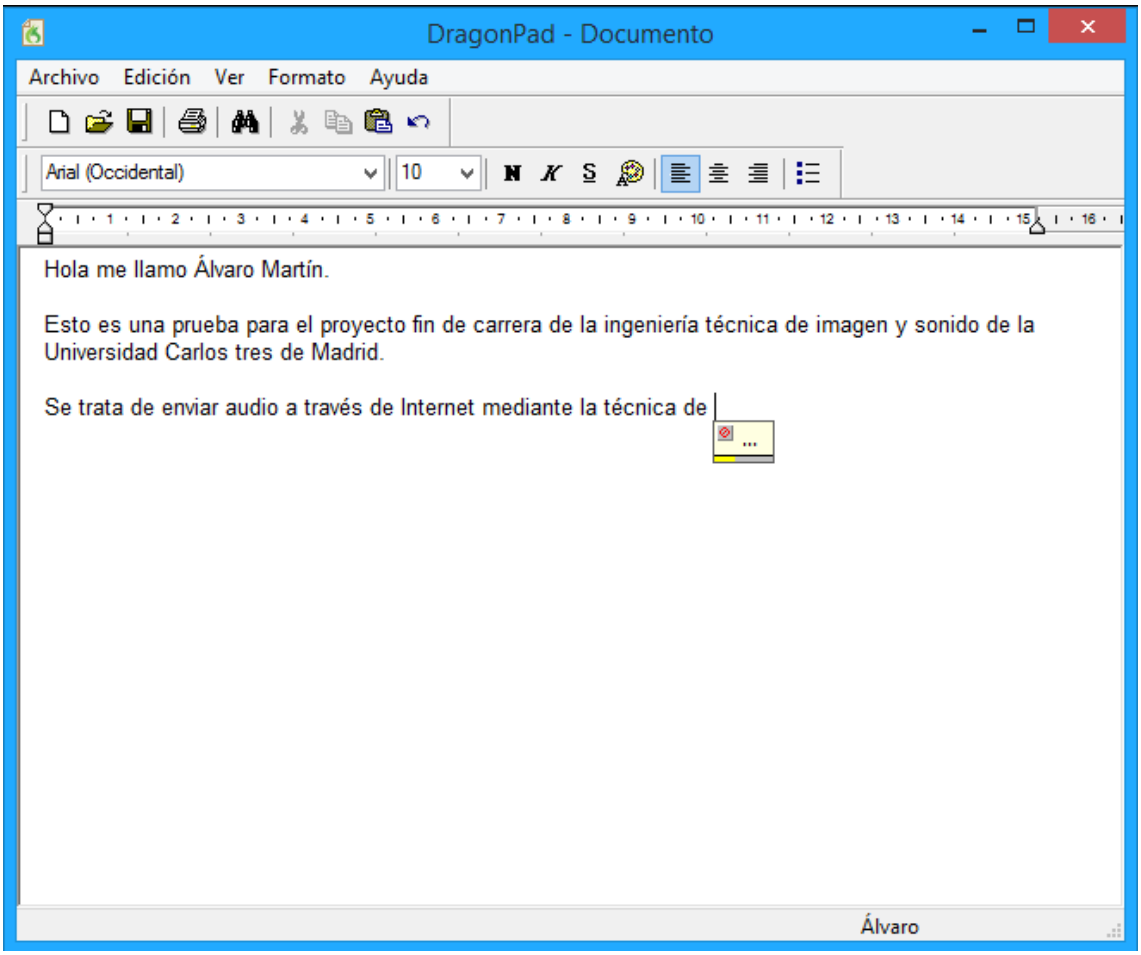

Figura 96. Ejemplo de dictado con DragonPad de DNS.

Se recomienda ser paciente y hablar a un ritmo pausado pues la transcripción de audio a texto no es un proceso inmediato.

Al finalizar el dictado, el usuario puede optar por guardar el resultado en un fichero de texto para su posterior uso o imprimirlo, las posibilidades son múltiples.